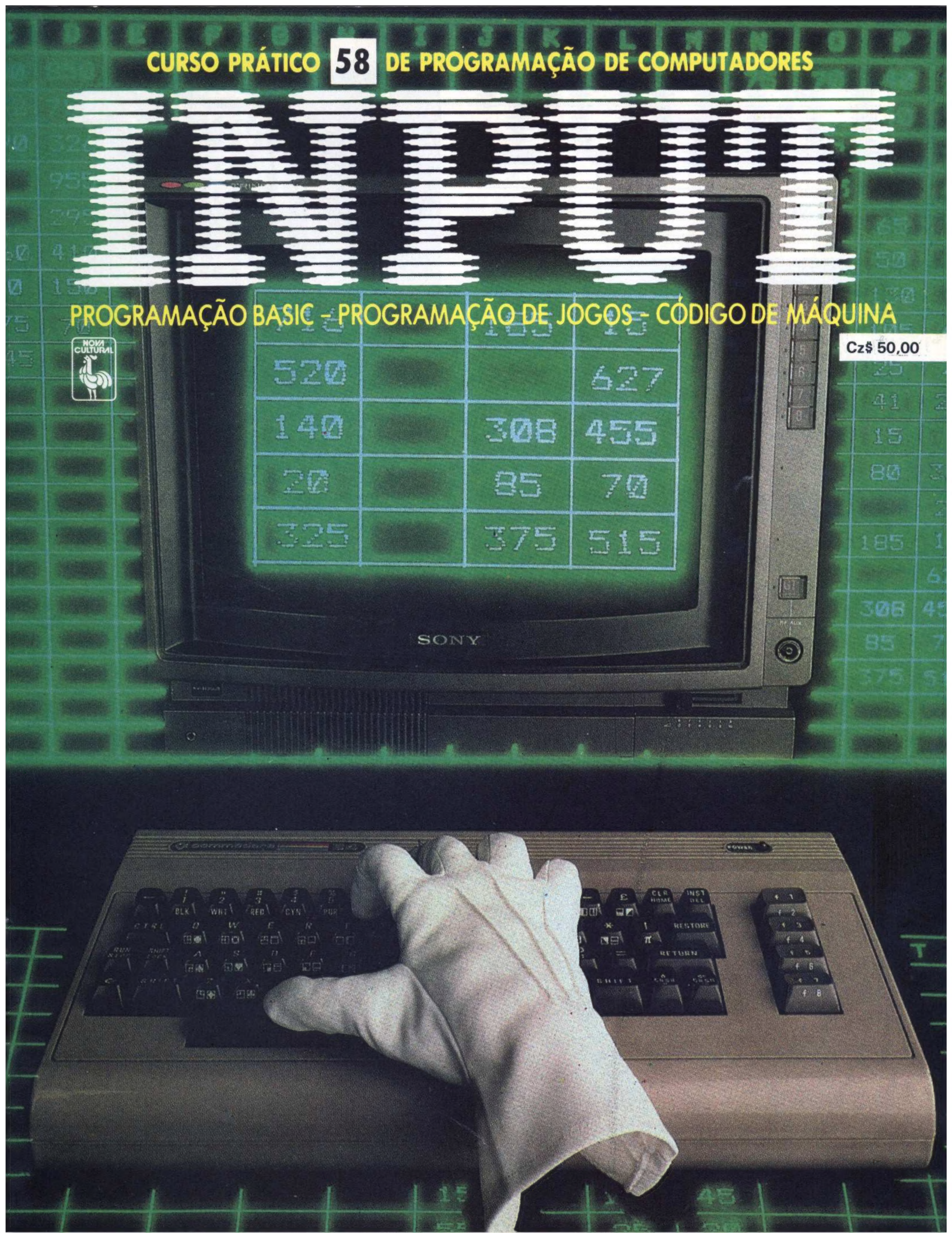
CURSO PRÁTICO 58 DE PROGRAMAÇÃO DE COMPUTADORES

INFO

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 50,00



INPUT

Vol. 4 N.º 58

NESTE NÚMERO

PROGRAMAÇÃO BASIC

MAIS SOBRE PÁGINAS GRÁFICAS

Retomando alguns pontos básicos sobre as técnicas de paginação gráfica. Limitações da memória. As páginas disponíveis em seu microcomputador. Páginas gráficas e a exigência de espaço. Armazenagem das páginas e sua recuperação. Animações gráficas 1141

CÓDIGO DE MÁQUINA

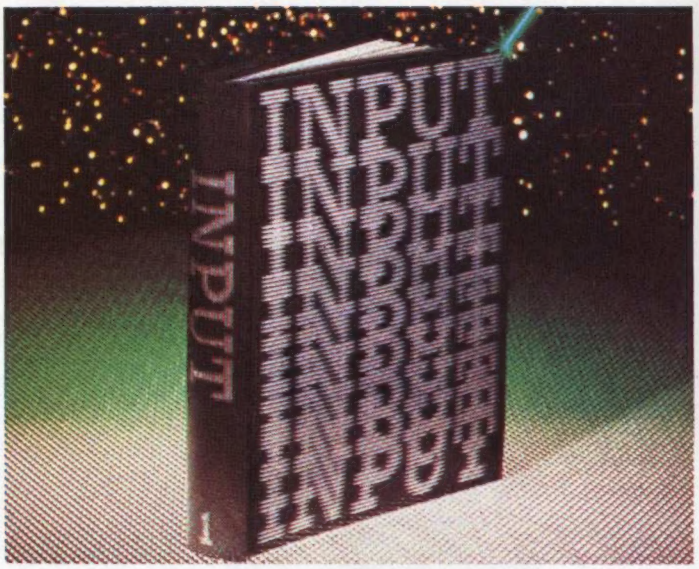
AVALANCHE: A ESCALADA

Animação do personagem. Início da caminhada. O momento de saltar. Onde Willie está pisando? Verificação dos prêmios. Imobilidade. Passos fatais. O último suspiro de Willie 1146

APLICAÇÕES

UMA PLANILHA ELETRÔNICA (3)

Instruções gerais sobre o uso da planilha. Como dar entrada às equações. Como fazer cópias absolutas e relativas. O uso de constantes. Comandos empregados. Como utilizar o programa. Instruções especiais para seu micro. Digite a última parte do programa 1155



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o n.º (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

NOVA CULTURAL

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editora de Texto: Ana Lúcia B. de Lucena
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/ Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3.º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda.
e impressa na Divisão Gráfica da Editora Abril S.A.

MAIS SOBRE PÁGINAS GRÁFICAS

- LIMITAÇÕES DA MEMÓRIA
- TÉCNICAS DE PAGINAÇÃO
- PÁGINAS GRÁFICAS E A EXIGÊNCIA DE ESPAÇO
- ANIMAÇÕES GRÁFICAS

Depois de examinar os princípios da paginação gráfica, vamos, agora, explorar suas possibilidades. Os programas deste artigo não deixarão dúvidas quanto à sua utilidade.

Como vimos no artigo da página 1096, a paginação gráfica — a técnica de mostrar várias telas gráficas em seqüência — apresenta um grande potencial em diversos tipos de aplicação. Além de seu emprego em animação computadorizada, as páginas gráficas podem ser úteis em áreas mais "sérias", tais como planilhas financeiras ou quaisquer outros projetos que exijam a rápida mudança de uma tela cheia de informações para outra.

Já tivemos um contato inicial com essa técnica. Agora, trataremos de examinar mais de perto suas possibilidades. Antes disso, no entanto, convém recapitular alguns pontos fundamentais. A paginação gráfica consiste, basicamente, em armazenar cada tela em uma parte da memória e depois recuperá-las uma a uma. Essas telas podem conter diferentes tipos de informação: gráficos de alta ou baixa resolução, textos, ou até uma combinação de ambos. Depois de armazenadas, elas podem ser chamadas em seqüência, sem que se perca tempo com o processo de montagem do desenho. O computador não precisará, portanto, executar uma série de funções demoradas, como a função **SIN**, por exemplo. Ele apenas transferirá dados de uma parte da memória para outra.

É claro que o processo de montagem do desenho continua sendo necessário em algum ponto, mas só uma vez para cada tela. Depois de prontas, elas podem ser recuperadas instantaneamente, sempre que você quiser.

LIMITAÇÕES DA MEMÓRIA

Cada página gráfica requer uma certa quantidade de memória. Essa quantidade varia de acordo com a complexidade do desenho — quanto mais cores forem usadas e quanto maior for a resolução gráfica, mais memória será exigida. Isso impõe severas limitações aos microcomputadores e, em alguns casos, temos mesmo que sacrificar um pouco a qualidade do desenho para obter mais páginas gráficas.

Tomando as medidas necessárias para economizar memória, você poderá conseguir bastante espaço para as páginas. Mas não se esqueça de que o próprio programa ocupará uma parte desse espaço. Portanto, antes de iniciar os trabalhos, calcule cuidadosamente a quantidade de memória que você ocupará com suas telas. Verifique se não é aconselhável baixar um pouco a resolução do desenho ou diminuir o número de cores. Com o planejamento, você não correrá o risco de precisar de mais espaço do que o disponível na RAM.

A técnica de paginação gráfica permite o acesso a cada tela na memória do computador. Dependendo do espaço disponível, pode-se definir previamente uma seqüência de oito ou mais telas. Para economizar espaço, lance mão de alguns pequenos truques. Por exemplo: em uma seqüência como a da página 1144, que mostra uma figura (o "homem-rabisco"), as páginas gráficas seriam exibidas na ordem 1, 2, 3, 4, 5, 1, 2, 3, 4, 5 e assim por diante. Porém, como você deve ter observado, existem dois pares muito semelhantes: o par 2 e 4 e o par 3 e 5. Ora, em uma situação onde qualquer espacinho está a prêmio, será bem melhor armazenar apenas uma figura de cada um desses pares. Durante a projeção das imagens, nem se notará a diferença e você terá economizado um bom espaço. Usando a seqüência 1, 2, 3, 2, 3, 1, você continuará com uma animação de cinco imagens, armazenando apenas três telas.

S

O Spectrum de 48 K pode manipular, no máximo, oito ou nove páginas gráficas diferentes, com apenas duas cores

(**INK** e **PAPER**). Esse micro também apresenta uma limitação que envolve o tamanho da tela. Se utilizarmos dois terços dela, cada página gráfica ocupará 4 K. Somando mais 2 K, exigidos pelo programa, chegamos logo ao limite máximo de memória. Levando em conta essas limitações, o programa que se segue trabalha com oito telas, mostrando uma estrada em perspectiva.

```
10 BORDER 0: PAPER 0: INK 7:
CLS
20 CLEAR 27999
30 GOSUB 170
40 LET srce=64: LET dest=110
50 FOR n=1 TO 20: PLOT RND*(
255),RND*(40)+130: NEXT n
60 FOR n=0 TO 7
70 FOR m=4 TO 21: PRINT AT m,
0;"
      ": NEXT m
80 GOSUB 260
90 GOSUB 220: LET dest=dest+
 16
100 NEXT n
110 LET srce=110: LET dest=64
120 FOR n=0 TO 7
130 GOSUB 220: LET srce=srce+
 16
140 PAUSE 4
150 NEXT n
160 GOTO 110
170 DATA 1,0,16,17,0,0,33,0,0,
237,176,201
180 FOR i=28000 TO 28000+11
190 READ byte: POKE i,byte
200 NEXT i
210 RETURN
220 POKE 28005,dest
230 POKE 28008,srce
240 RAND USR 28000
250 RETURN
260 PLOT 0,120: DRAW 255,0
270 PLOT 118,120: DRAW -118,-
80: PLOT 138,120: DRAW 117,-50
280 FOR j=1 TO 3: READ x,y,a,b
: PLOT x,y: DRAW a,b: NEXT j
290 READ x,y,a,b,c,d
300 PLOT x,y: DRAW a,b: DRAW c
,d
310 RETURN
320 DATA 128,120,1,-1,140,105,
3,-3,138,120,0,2,118,140,10,-5
,10,5,130,118,1,-1,160,80,6,-6
,143,118,0,7,118,138,10,-3,10,
3
330 DATA 133,114,1,-1,198,30,8
,-8,160,112,0,15,118,136,10,-1
,10,1,140,105,4,-4,128,120,1,-
1,184,105,0,30,118,132,10,3,10
,-3,160,80,6,-6,130,118,1,-1
340 DATA 220,90,0,50,118,134,
10,1,10,-1,198,30,8,-8,133,114
,1,-1,118,120,0,4,118,136,10,-
1,10,1
350 DATA 128,120,1,-1,140,105,
4,-4,80,100,0,30,118,138,10,-3
,10,3,130,118,1,-1,160,80,6,-6
,5,55,0,100,118,139,10,-4,10,4
```

O programa começa inicializando a tela em preto e branco e definindo o **RAMTOP** em 27999. A rotina das linhas 170, 180, 190, 200 e 210 coloca

uma pequena rotina em linguagem de máquina acima da RAM acessível. Essa rotina, muito rápida, será a responsável pela transferência do bloco de informações da tela para a memória, à medida que as imagens vão sendo construídas e, posteriormente, irá também recuperá-las, trazendo-as de volta à tela. A linha 40 estabelece os valores dos bytes mais significativos das variáveis **srce** e **dest**, que contêm, respectivamente, o endereço da área a ser transferida e o do seu destino.

A primeira parte da rotina dos gráficos, que começa na linha 50, simplesmente faz desenhos em posições aleatórias da tela. Esses desenhos são fixos e, apesar de aparecerem em todas as páginas, não são refeitos a cada vez, pois não estão incluídos na seqüência principal de figuras. A linha 170 se encarrega de limpar (sobrepondo espaços) a parte de baixo de cada tela, sem apagar as estrelas que estão no alto. Um laço de oito telas já estará, então, começando (linha 60). A rotina de gráficos desenha o horizonte (linha 260), os lados da estrada (linha 270), a própria estrada e os postes (linha 280) e uma ave em vôo (linha 290) — sempre lendo as informações que estão nas linhas **DATA** ao final do programa (linha 320 em diante).

Retornando da rotina de desenho, o programa é desviado para uma subrotina de **POKE**, na linha 220. Esta copia os 4 K de cada tela em um local apropriado da memória. O endereço **dest** é incrementado com o valor 16 no seu byte mais significativo (16*255 = 4 K), a fim de criar o espaço para a próxima página gráfica. O programa volta, mais uma vez, para a rotina de desenho e todo o processo se repete oito vezes, sendo que, a cada vez, a figura é feita uma posição adiante.

Em seguida, o programa chama as oito telas, sucessivamente, dando o efeito de animação. Para isso, o endereço **dest** torna-se o novo **srce**, a partir do qual a rotina da linha 220 à 250 chama cada página gráfica.

O TRS-Color apresenta evidente vantagem sobre os demais microcomputadores, em virtude de seu poderoso comando **PCOPY**, que permite que o usuário manipule páginas gráficas sem muitas complicações. No exemplo que se segue, optamos pela modalidade gráfica **PMODE 3.** Cinco páginas gráficas, ocupando três quartos da tela, serão usadas para se obter uma animação.

```
10 PCLEAR 4:PMODE 3:CLEAR 40,92
15
20 SCREEN 1,0:FOR K=0 TO 4:PCLS
30 CIRCLE(127,120),20,4,1,.17,.
55:LINE(110,116)-(127,120),PSET
:LINE-(132,136),PSET:PAINT(122,
135),2,4
40 DRAW"BM137,136S8F6D10L9U15L4
D19R17U14E2NE6L8":PAINT(150,150
),3,4:DRAW"C3BRR4C4"
50 DRAW"BM110,116S16L14U10R21D3
RU5L24D14R15":PAINT(90,120),3,4
60 COLOR 3:FOR L=0 TO 5-K:LINE(
148-L,146-L)-(156+L,146-L),PSET
:NEXT
70 DRAW"BM141,"+STR$(86+K)+"C3S
4F2G2H3E2D4":DRAW"BM141,"+STR$(
INT(88+1.5*K*K))+"D2F2DL4UE2D4"
80 DRAW"BM"+STR$(INT(141+1.8*K)
)+","+STR$(127+5*K)+"H3E3F2G2DU
4G2"
90 COLOR 2:FOR L=0 TO 5:LINE(11
0-L*10-K*2,117)-(110-L*10-K*2,1
23),PSET:NEXT
100 FOR L=0 TO 7:LINE(54+L*10+K
*2,75)-(54+L*10+K*2,69),PSET:NE
XT
110 FOR L=0 TO 3:LINE(48,115-L*
10-K*2)-(52,116-L*10-K*2),PRESE
T:NEXT
120 IF K=0 THEN DRAW"BM110,124C
3H2UE2F2DG2U4"
130 COLOR 4:FOR L=0 TO 2:A=ATN(
1)*(L*60-K*12)/45:LINE(127-18*S
IN(A),120-18*COS(A))-(127+18*SI
N(A),120+18*COS(A)),PSET:NEXT
140 A=ATN(1)*(8+K*12)/45:DRAW"B
M"+STR$(INT(127-18*SIN(A)))+","
+STR$(INT(120+18*COS(A)))+"C3E2
UH2G2DF2U6C4"
150 FOR L=2 TO 4:PCOPY L TO 4+K
*3+L:NEXT L,K
160 FOR L=1 TO 5:FOR K=2 TO 4:P
COPY K+L*3+1 TO K:NEXT K,L:GOTO
 160
```

O programa começa reservando quatro blocos (1,5 K cada) para as informações da tela, pois, como selecionamos a modalidade **PMODE 3**, cada tela requisitará 6 K de memória.

Até agora utilizamos 6 K. Como o BASIC e a tela de teste ocuparão mais 1,5 K, ficaremos com 25 K de memória RAM aproveitáveis. Usando páginas inteiras que requerem 6 K cada uma, poderíamos contar com mais de quatro telas de páginas gráficas (25 K divididos por 6) — mas isto deixaria muito pouco espaço para o programa.

Se limitássemos o desenho a três quartos da tela, cada página iria ocupar apenas 4,5 K de memória. Utilizaríamos, assim, cinco páginas, deixando livres 2 K para o programa. Não haveria, entretanto, espaço para o sistema de operação de disco.

Depois de estabelecer **PMODE 3** — quatro cores com uma resolução de 128 por 192 pontos —, a primeira linha do programa limpa a parte menos importante de armazenamento de strings acima do endereço 9215. Para suas próprias rotinas, você deverá definir essa parte por tentativa e erro.

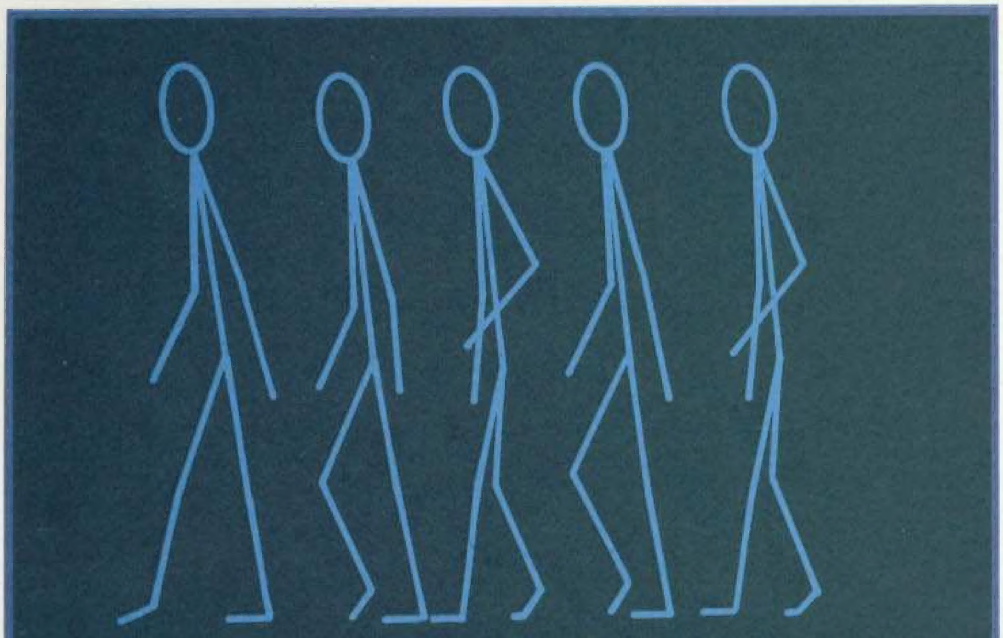
A segunda linha continua com o processo de inicialização, definindo o modo de resolução da tela e a cor. Aqui se inicia também o laço de desenho, cujo primeiro comando é um **PCLS.**

A rotina de desenho que se segue ocupa grande parte do programa. A linha 30 constrói e pinta o corpo da bomba de movimento contínuo, que forma a base da figura. A linha 40 monta o funil e o preenche com cor, enquanto a linha 50 faz o mesmo com o cano. As linhas 60, 70 e 80 encarregam-se da queda da água. As linhas 90, 100 e 110 desenham as listas que ajudam a dar a impressão de que a água está em movimento. Outros detalhes da bomba — como a rotação da pá — são executados pelas linhas 120, 130 e 140.

A linha 150 copia os três quartos da tela na memória — mais precisamente na área definida pela linha 10 — e o programa volta para a rotina de desenho, para que sejam montadas as outras páginas gráficas. Cada uma delas é guardada na memória quando o programa alcança, novamente, a linha 150.

Depois que todas as páginas estiverem armazenadas, o programa entra na rotina de animação (linha 160). Esta chama todas as telas, em seqüência, até o programa ser interrompido.

Quando usamos um comando gráfico do tipo **LINE**, por exemplo, o computador "rabisca" o traço pedido sobre a tabela de padrões da VRAM. Portanto, se fizermos uma cópia dessa tabela na RAM, podemos transferi-la posteriormente para a VRAM e o desenho se-



Entre as cinco figuras, existem dois pares muito semelhantes: o par 2 e 4 e o par 3 e 5. Podemos, sem maiores prejuízos, armazenar só três figuras e repetir duas delas — uma de cada par — na seqüência. Obteremos, desse modo, uma animação de cinco imagens utilizando apenas três telas. Este é um dos vários recursos empregados para economizar memória.

rá imediatamente refeito. Poderíamos também armazenar a tabela de cores, mas isso acrescentaria 6144 bytes aos 6144 bytes da tabela de padrões, o que nos limitaria ao uso de, no máximo, uma página gráfica. Para que nossa tela não fique sem cor, carregaremos a tabela de cor com um número específico que definirá as cores de fundo e de frente. Nosso desenho (um beija-flor) terá, portanto, apenas duas cores. Em compensação, economizamos o suficiente para três páginas gráficas.

Para transferir os dados da VRAM para a RAM, recorreremos a uma rotina da ROM chamada **LDIRMV**. Para fazer o contrário, usaremos uma outra rotina, a **LDIRVM**. Finalmente, para carregar a tabela de cores com um byte específico, utilizaremos a rotina **FILVRM**, que preenche qualquer área da VRAM com um determinado valor. O acesso a essas rotinas requer a introdução de alguns valores nos registradores da máquina. Por isso, construiremos uma pequena rotina em linguagem de máquina para chamar cada rotina da ROM.

```
10 CLS
20 CLEAR200,40960!
30 DEFUSR=40960!
40 DEFUSR1=40973!
50 DEFUSR2=40986!
60 FOR R=0 TO 37
70 READ A
80 POKE (40960!+R),A
90 NEXT
100 SCREEN2
110 X1=112:Y1=67:N=16
120 GOSUB 430
130 A=USR(0)
140 SCREEN2
150 X1=115:Y1=83:N=19
160 GOSUB 430
170 POKE 40964!,0
180 POKE 40965!,185
190 A=USR(0)
200 SCREEN 2
210 X1=158:Y1=182:N=17
220 GOSUB 430
230 POKE 40964!,0
240 POKE 40965!,209
250 A=USR (0)
260 POKE 40990!,1*16+11
270 A=USR2(0)
280 FOR I=1 TO 3:ON I GOSUB 310
,350,390:NEXT
290 FOR I=3 TO 1 STEP -1:ON I G
OSUB 310,350,390:NEXT
300 GOTO 280
310 POKE 40974!,0
320 POKE 40975!,161
330 A=USR1(0)
340 RETURN
350 POKE 40974!,0
360 POKE 40975!,185
370 A=USR1(0)
380 RETURN
390 POKE 40974!,0
400 POKE 40975!,209
410 A=USR1(0)
420 RETURN
430 FOR I=1 TO N
440 READ X2
450 READ Y2
460 LINE(X1,Y1)-(X2,Y2)
470 LET X1=X2:LET Y1=Y2
480 NEXT I
490 RETURN
500 DATA 33,00,00,17,00,161,01,
00,24,205,89,00,201
510 DATA 33,00,161,17,00,00,01,
00,24,205,92,00,201
520 DATA 33,00,32,62,00,01,00,2
4,205,86,00,201
530 DATA 86,3,79,67,90,86,192,3
,176,64,141,99,198,166,170,186
540 DATA 115,186,118,138,86,118
,12,191,48,139,51,109,67,90,90,
86
550 DATA 96,64,54,48,83,86,115,
83,160,80,240,102,185,115
560 DATA 144,113,160,144,192,14
7,179,169,144,185,115,148
570 DATA 86,118,12,191,48,139,5
1,109,67,90,83,86
580 DATA 182,160,224,121,166,12
8,105,83,90,128,160,191
590 DATA 150,116,105,83,67,90,5
1,109,48,139,10,190
600 DATA 89,118,90,128,87,137,8
3,176,70,134
```

A linha 20 reserva uma área da memória para as rotinas em linguagem de máquina. A linha 30 define o início da rotina que irá chamar a **LDIRMV** e a linha 40, o início da rotina que acessa a **LDIRVM**. A linha 50 faz o mesmo para a rotina **FILVRM.**

O laço entre as linhas 60 e 90 lê as linhas **DATA** 500, 510 e 520, que contêm as rotinas — na ordem em que foram citadas —, colocando-as na memória do microcomputador.

A rotina de desenho localiza-se entre as linhas 100 e 270. A cada página, o computador inicializa as variáveis **X1**, **Y1** e **N**, que guardam as coordenadas iniciais do desenho e o número de pontos a serem lidos. O programa é então desviado para a sub-rotina entre as linhas 430 e 490, que lê as informações contidas nas linhas 530 a 600 e traça as retas. Assim que a figura estiver completa, a rotina que armazena a tela na RAM é chamada.

O processo é o mesmo para as outras duas páginas, com uma diferença: dois comandos **POKE** irão alterar a sub-rotina de armazenamento. Esta continha, inicialmente, o endereço a partir do qual a primeira tela seria guardada na RAM. É claro que não podemos usar esse mesmo endereço para as duas outras páginas gráficas. O papel dos comandos **POKE** é, portanto, fazer a modificação necessária para o início da segunda e da terceira páginas.

Depois que todas as telas já tiverem sido armazenadas, a linha 260 determina o valor do byte que preencherá a ta-

bela de cores, colocando-o na rotina que acessa a **FILVRM**. O valor 1 é a cor de frente — preto — e o valor 11 é a cor de fundo — amarela. A linha 270 chama essa rotina.

Em seguida chega-se à rotina encarregada de alternar as telas. Os dois laços (linhas 280 e 290) determinam uma seqüência de telas do tipo 1, 1, 2, 3, 3, 2, 1, 1, fazendo com que um beija-flor em vôo permaneça mais tempo com as asas para cima e para baixo do que na posição intermediária.

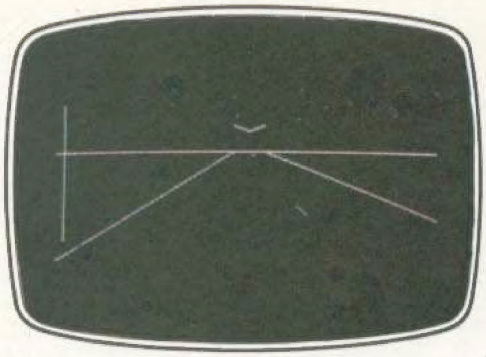
Conforme o valor da variável **I**, o programa é desviado para uma sub-rotina. Nesta, um par de comandos **POKE** modifica a rotina que busca os dados na RAM. Esses comandos definem os endereços a partir dos quais a memória será transferida para a tela.

Se você quiser construir suas próprias animações, elimine a sub-rotina da linha 430, assim como todas as linhas que a chamam. Além disso, apague também a linha **DATA** 530 e as que a seguem e coloque seus comandos gráficos a partir da linha que inicializava as variáveis, tendo o máximo cuidado para não provocar sobreposições.

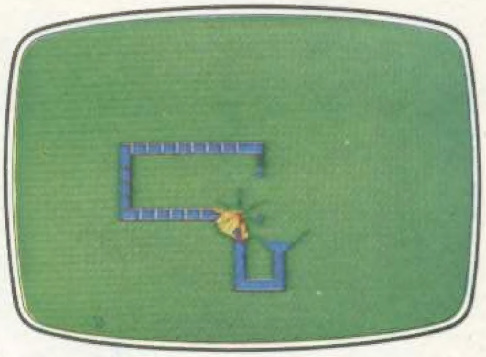
Como você deve saber, os micros da linha Apple oferecem ao usuário duas páginas gráficas prontas para serem utilizadas. Por meio dos comandos usuais do BASIC, não podemos escrever em uma das páginas sem que ela apareça no vídeo. Se isto fosse possível, eliminaríamos, em uma animação, aquele efeito desagradável provocado pelos comandos **HGR** e **HGR2** para limpar a tela — enquanto mostrássemos uma tela, estaríamos simultaneamente limpando e escrevendo na outra. Em seguida, bastaria trocar as duas e repetir o processo até o fim da animação. Teríamos, assim, uma velocidade moderada, mas não haveria um limite de páginas gráficas.

O próximo programa ilustra o desenvolvimento desse processo com a animação de uma figura — o homem-rabisco —, comentada anteriormente.

```
10  POKE  - 16304,0:X = 200
20  HGR : HGR2 : HCOLOR= 3
30  POKE 230,32: GOSUB 240
40  POKE  - 16300,0
50  POKE 230,64: GOSUB 280
60  POKE  - 16299,0
70  POKE 230,32: HCOLOR= 0: GOS
UB 240: HCOLOR= 3: GOSUB 320
80  POKE  - 16300,0
90 P1 = 32:P2 = 64:M1 =  - 1630
0:M2 =  - 16299
100  POKE 230,P2: HCOLOR= 0: GO
SUB 280:X = X - 15: HCOLOR= 3:
GOSUB 280
110  IF X < 30 THEN  STOP
120  POKE M2,0
130  POKE 230,P1:X = X + 15: HC
OLOR= 0: GOSUB 320:X = X - 15:
HCOLOR= 3: GOSUB 320
140  POKE M1,0
150  POKE 230,P2: HCOLOR= 0: GO
SUB 280: HCOLOR= 3: GOSUB 240
160  POKE M2,0
170  POKE 230,P1: HCOLOR= 0: GO
SUB 320: HCOLOR= 3: GOSUB 280
180  POKE M1,0
190  POKE 230,P2: HCOLOR= 0: GO
SUB 240: HCOLOR= 3: GOSUB 320
200  POKE M2,0
210 TR = P1:P1 = P2:P2 = TR
220 TR = M1:M1 = M2:M2 = TR
230  GOTO 100
240  HPLOT X - 16,136 TO X - 9,
133 TO X,94 TO X + 8,136 TO X,1
36
250  HPLOT X,94 TO X - 3,58 TO
X - 9,50 TO X,48 TO X - 3,58
260  HPLOT X - 10,97 TO X - 4,8
2 TO X - 3,58 TO X + 9,98
270  RETURN
280  HPLOT X - 15,136 TO X - 11
,132 TO X - 10,114 TO X - 1,97
TO X + 7,136 TO X,136
290  HPLOT X - 1,97 TO X - 7,61
 TO X - 13,50 TO X - 4,48 TO X
- 7,61
300  HPLOT X - 12,97 TO X - 6,8
5 TO X - 7,61 TO X + 1,82 TO X
+ 2,97
310  RETURN
320  HPLOT X - 15,136 TO X - 8,
136 TO X - 5,114 TO X - 1,94 TO
 X - 5,114 TO X + 3,132 TO X -
3,136
330  HPLOT X - 1,94 TO X - 5,61
 TO X - 12,50 TO X - 3,48 TO X
- 5,61
340  HPLOT X - 7,98 TO X - 3,61
 TO X + 5,78 TO X - 8,90
350  RETURN
```



A estrada sem fim no Spectrum.



TRS-Color: bomba moto-contínua.

ANIMAÇÃO GRÁFICA

As instruções para as telas de números 1, 2 e 3 se encontram, respectivamente, a partir das linhas 240, 280 e 320. Elas serão chamadas na ordem 1, 2, 3, 2, 3, 1. Se quisermos mostrar um desenho na página 2, por exemplo, teremos que executar as etapas do processo descrito em seguida, enquanto a página 1 está na tela:

- habilitar a escrita na página 2
- apagar a página 2
- escrever na página 2
- mostrar a página 2 na tela

Para habilitar a escrita em uma página, é necessário colocar um determinado valor no endereço 230 por meio de um comando **POKE**. Se esse valor for 32, qualquer comando gráfico será executado na página 1; se for 64, será acessada a página 2.

Para apagar uma página gráfica, temos simplesmente de descolorir a figura, utilizando a mesma rotina que a montou, com **HCOLOR = 0**.

Empregamos também o comando **POKE** para colocar uma página na tela. Se esse comando for dado no endereço −16300, será mostrada a página 1; se for dado no endereço −16299, será mostrada a página 2. Em ambos os casos, porém, deve haver um comando **POKE** no endereço −16304; caso contrário, será mostrada a página de texto.

A fase de inicialização localiza-se entre as linhas 10 e 90, onde ocorrem as primeiras impressões. A variável **X** controla a posição horizontal da figura na tela e será incrementada com o valor 15 a cada passo.

Entre as linhas 100 e 230 situa-se o laço que irá controlar o movimento do homem-rabisco até o fim da animação. Na linha 100 e na linha 130, subtrai-se 15 dessa variável para que seja apagada a imagem que foi feita antes do incremento.

As linhas 210 e 220 promovem a troca de valores entre as variáveis **P1**, **P2**, **M1** e **M2**. Estas definem as páginas em que serão feitos os desenhos e as páginas que serão mostradas. A troca é necessária para que não se perca a alternância de telas.

AVALANCHE: A ESCALADA

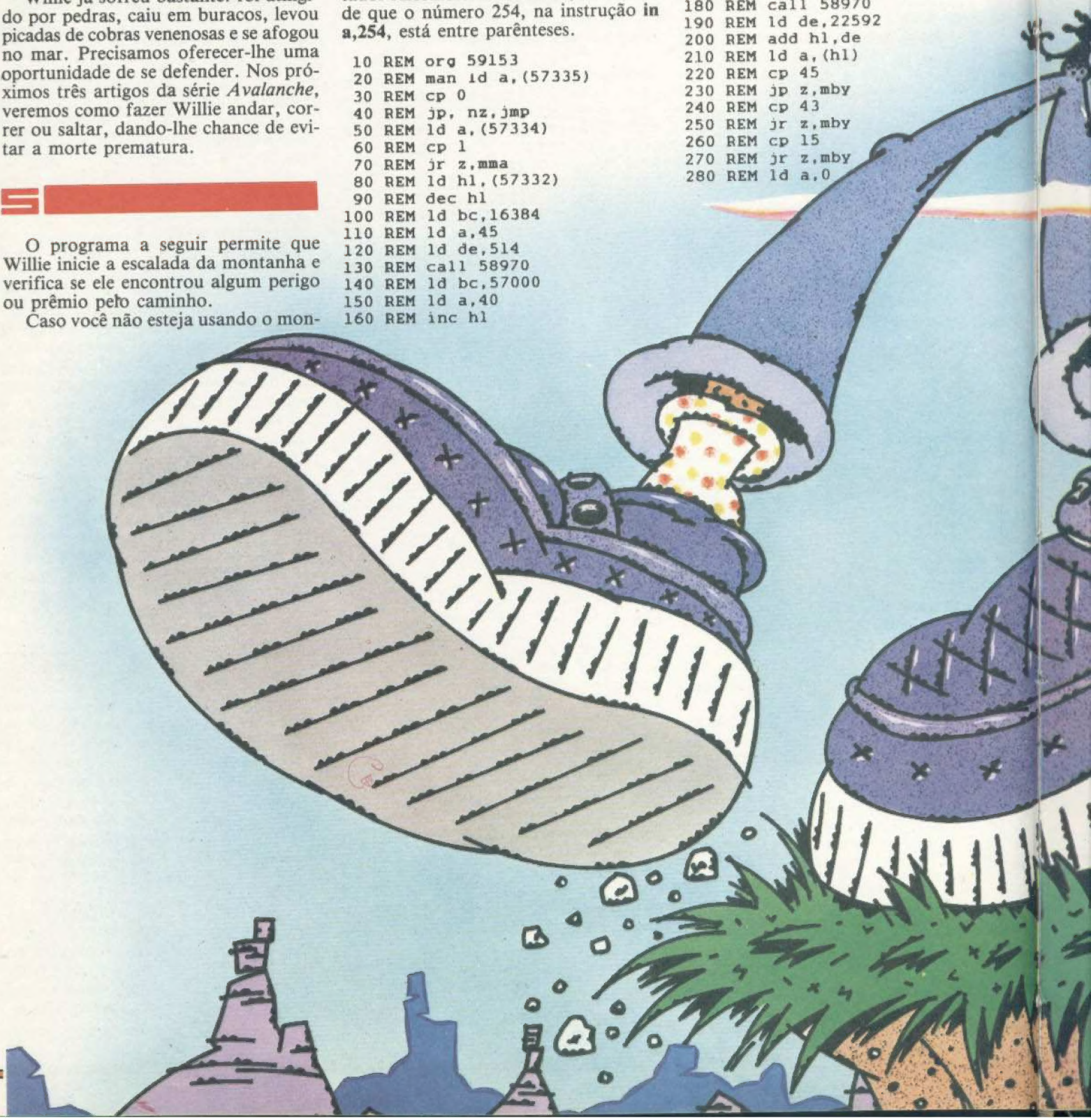
Até agora, nosso personagem permaneceu indefeso diante dos perigos que o cercam. Vamos dar-lhe a chance de escapar, fazendo com que escale a montanha, corra e salte.

Willie já sofreu bastante: foi atingido por pedras, caiu em buracos, levou picadas de cobras venenosas e se afogou no mar. Precisamos oferecer-lhe uma oportunidade de se defender. Nos próximos três artigos da série *Avalanche*, veremos como fazer Willie andar, correr ou saltar, dando-lhe chance de evitar a morte prematura.

O programa a seguir permite que Willie inicie a escalada da montanha e verifica se ele encontrou algum perigo ou prêmio pelo caminho.

Caso você não esteja usando o montador Assembler de INPUT, lembre-se de que o número 254, na instrução **in a,254**, está entre parênteses.

```
10 REM org 59153
20 REM man ld a,(57335)
30 REM cp 0
40 REM jp, nz,jmp
50 REM ld a,(57334)
60 REM cp 1
70 REM jr z,mma
80 REM ld hl,(57332)
90 REM dec hl
100 REM ld bc,16384
110 REM ld a,45
120 REM ld de,514
130 REM call 58970
140 REM ld bc,57000
150 REM ld a,40
160 REM inc hl
170 REM ld de,258
180 REM call 58970
190 REM ld de,22592
200 REM add hl,de
210 REM ld a,(hl)
220 REM cp 45
230 REM jp z,mby
240 REM cp 43
250 REM jr z,mby
260 REM cp 15
270 REM jr z,mby
280 REM ld a,0
```



- ANIMAÇÃO DO PERSONAGEM
- O MOMENTO DE SALTAR
- INÍCIO DA CAMINHADA
- ONDE WILLIE ESTÁ PISANDO?
- VERIFICAÇÃO DOS PRÊMIOS
- IMOBILIDADE
- PASSOS FATAIS
- O ÚLTIMO SUSPIRO

```
290 REM in a,254
300 REM bit 2,a
310 REM jr nz,mft
320 REM ld b,1
330 REM bit 3,a
340 REM jr nc,mlj
350 REM ld b,129
360 REM mlj ld a,b
370 REM ld (57335),a
380 REM jr mct
390 REM mft bit 3,a
400 REM jr nz,mct
410 REM ld a,1
420 REM ld(57334),a
430 REM mct ld hl,(57332)
440 REM ld de,191
450 REM sbc hl,de
460 REM jr nc,mor
470 REM ld a,1
480 REM ld (57336),a
490 REM mor ret
500 REM mma ld de,3
510 REM ld hl,1548
520 REM call 949
530 REM ld hl,(57332)
540 REM ld de,22561
550 REM add hl,de
560 REM ld a,(hl)
570 REM cp 43
580 REM jr z,mby
590 REM cp 44
600 REM jr z,mts
610 REM cp 42
620 REM jr z,mby
630 REM ld de,32
640 REM add hl,de
650 REM ld a,(hl)
660 REM cp 15
670 REM jr z,mby
680 REM cp 45
690 REM jr z,mby
700 REM cp 43
710 REM jr z,mby
720 REM ld hl,(57332)
730 REM ld a,40
740 REM ld bc,57016
750 REM ld de,514
760 REM call 58970
770 REM inc hl
780 REM ld (57332),hl
790 REM mts ld a,0
800 REM ld (57334),a
810 REM ret
820 REM mby ld a,2
830 REM ld (57336),a
840 REM ret
850 REM jmp ret
```

Você pode testar essas linhas mesmo que ainda não tenha na memória as outras duas rotinas de movimentação de Willie. Uma instrução **ret**, que será apagada mais tarde, foi colocada no fim da rotina que acabamos de listar. Em conseqüência, se você chamar uma rotina inexistente, o programa apenas retorna e nenhum erro ocorre.

QUAL É O MOVIMENTO?

A variável na posição de memória 57335 informa se Willie irá ou não pular. O conteúdo desta posição é carregado no acumulador e comparado com 0. Se não for igual a 0, Willie irá pular. O processador salta, então, para a rotina **jmp**, que ainda não foi publicada. Como, em seu lugar, encontra apenas uma instrução **ret**, volta para o local onde a rotina foi chamada.

Se Willie não vai pular (o conteúdo de 57335 é 0), o processador continua a execução da rotina. O conteúdo da posição de memória 57334 é carregado no acumulador. Essa posição contém a chamada *variável da caminhada*.

Existem duas figuras para Willie — numa delas, ele está com as pernas juntas e, na outra, com as pernas abertas. Enquanto nosso personagem não se locomove, a primeira figura é impressa continuamente no mesmo lugar. Mas, quando ele está andando, as duas figuras são impressas alternadamente. A variável da caminhada informa ao processador se Willie deve andar uma posição — nesse caso, a figura adequada é a que tem as pernas abertas.

A instrução **cp 1** verifica qual figura será impressa. Se for a de valor 1 — que tem as pernas abertas —, a instrução **jr z,mma** manda o processador para a rotina que faz Willie andar e imprime essa figura. Se for a de valor 0 — que tem as pernas fechadas —, o processador continua.

O HOMEM INVISÍVEL

Ao andar, Willie abre as pernas e se desloca uma posição à frente, onde aparece com as pernas juntas. Portanto, o algoritmo que produz esse efeito imprime, em seqüência, as figuras 0, 1 e 0.

Como a figura que tem as pernas abertas ocupa duas posições adjacentes e a que tem as pernas fechadas, só uma, obtém-se um movimento bem suave.

Para imprimir a figura 0, é preciso apagar a figura anterior — do contrário, teremos várias partes de Willie pela tela. A instrução **ld hl,(57332)** carrega a posição de Willie — armazenada nos endereços 57332 e 57333 — no par HL. Esse valor é decrementado para voltar uma posição.

O par BC é então carregado com

16384. Este é o endereço do topo da tela, onde se encontra o padrão de céu. A é carregado com 45 e DE, com 514. A rotina de impressão de blocos em 58970 é chamada em seguida. Como você deve se lembrar, o par HL, nessa rotina, contém a posição na tela, e o par BC, o apontador de dados. A, por sua vez, especifica a cor — 45 é ciano sobre ciano — e o par DE fixa o tamanho do bloco. O número 514 fornece um bloco de dois por dois caracteres. D contém o número de linhas e E, o número de colunas — 2 × 256 + 2 = 514.

Assim, quando é chamada, essa rotina imprime um bloco dois por dois de céu na posição imediatamente anterior àquela onde você irá imprimir Willie de novo. Em outras palavras, ela apaga o velho Willie que, com suas pernas abertas, ocupa quatro caracteres.

IMOBILIDADE

Agora precisamos imprimir o novo Willie com as pernas juntas. Os dados para isso começam em 57000. Logo, o par BC é carregado com 57000. O acumulador A é carregado com 40, código de azul sobre fundo ciano, a cor de Willie.

O par HL, anteriormente decrementado, volta a ser incrementado para apontar a posição do personagem. O par DE é carregado com 258 — dando um bloco de um por dois (1 × 256 + 2 = 258), o espaço que Willie ocupa com as pernas juntas. A rotina de impressão de blocos é chamada para imprimir Willie.

ONDE WILLIE ESTÁ PISANDO?

Como nosso personagem se moveu uma posição à frente, convém verificar onde ele está pisando. Se for numa cobra, num buraco ou na água, o processador deve ser informado de sua morte.

Em primeiro lugar, verifique a cor do caractere que está sob os pés de Willie. O arquivo de cores começa em 22528. O valor contido em HL corresponde à posição da tela ocupada pela cabeça de Willie. Mas, como você quer a cor do caractere que está sob seus pés, precisará adicionar o valor 64.

Por isso, 22592 é carregado no par de registros DE e somado ao par HL. O resultado da instrução **add hl,de** é sempre colocado em HL. O conteúdo da posição da memória que HL aponta no momento é colocado em A. O acumulador passa a conter a cor do caractere que está sob os pés de Willie.

Esse valor é comparado com 45 — ciano sobre ciano, a cor do céu que preenche os buracos. Se for 45, a instrução **jr z,mdy** manda o processador para uma rotina curta que faz Willie morrer. Se não há um buraco debaixo de Willie, o conteúdo do acumulador é comparado com 43 — magenta sobre ciano, a cor da cobra — e com 15 — branco sobre azul, a cor do mar. Se qualquer um desses valores estiver presente, o processador salta para a rotina **mdy** e elimina Willie. Caso contrário, o processador continua a execução da rotina — Willie está salvo.

HORA DE PULAR

Nesta parte da rotina, a ação se encerra, efetivamente, com a impressão de Willie com as pernas juntas, uma posição à frente. Se ele não morreu, o passo seguinte consiste em verificar quando ele irá se mover de novo. Para isso, empregamos o comando **in**.

Antes, porém, A é carregado com 0 — o que significa que o teclado inteiro será analisado e que qualquer conjunto de teclas poderá ser utilizado para controlar os movimentos de Willie. Embora na página de instruções M e N tenham sido especificadas, outras combinações — tais como J e K ou U e I — são apropriadas para fazer nosso personagem pular e andar.

O comando **in** é usado para efetuar uma busca na porta 254. A instrução **bit 2,a** analisa o bit dois do número encontrado para verificar se a tecla M — ou J ou U — foi pressionada. Todos os bits equivalentes às teclas têm normalmente o valor 1, mas assumem o valor 0 quando a tecla é pressionada. Assim, se M foi pressionada, o bit dois tem valor 0 e o processador ignora a instrução **jr nz.** Caso contrário, ele salta para a rotina **mft**.

Se a tecla foi pressionada e Willie deve pular, B é carregado com 1. Em seguida, o bit três é testado do mesmo modo. Se N não foi pressionada e o bit três tem valor 1, a instrução **jr nz** faz o processador saltar a próxima instrução. Mas, se a tecla foi pressionada, a instrução **jr nz** não tem efeito e B é carregado com 129.

O significado dos números 1 e 129, que utilizamos para carregar B, será explicado quando apresentarmos as seções do programa que executam o salto vertical e o salto à frente.

O processador transfere o conteúdo de B para A e o carrega em 57335. Este, como você deve se lembrar, corresponde à posição de memória que o processador irá examinar no início da rotina para ver se Willie pulará ou não.

A instrução **jr mct** faz o processador saltar a próxima rotina.

WILLIE VAI ANDAR?

Se não houve pressão sobre M, o processador vai para a rotina **mft**. Ela verifica se apenas a tecla N foi pressionada — ou seja, se Willie, em vez de pular, simplesmente irá andar.

Para ver se a tecla N foi pressionada, o bit três é testado de novo. Note que o valor da porta 254 ainda está no acumulador — o processador pulou para este ponto da rotina depois que o bit dois foi testado.

Se N não foi pressionada, **jr nz** salta a rotina seguinte. Caso contrário, A é carregado com 1 e esse valor é armazenado no endereço 57334, onde se encontra a variável da caminhada (que examinamos no início da rotina para saber qual das figuras de Willie seria impressa na tela). Um valor 1 nessa posição indica que Willie deve ser impresso com as pernas abertas — em outras palavras, ele está andando.

PRÊMIOS

Há apenas mais uma verificação a fazer: nosso personagem alcançou algum prêmio?

A posição de Willie é armazenada nos endereços 57332 e 57333. O conteúdo desses endereços é carregado no par HL e o valor 191 — correspondente à posição de tela onde o prêmio foi impresso — é colocado em DE.

O conteúdo desse par de registros é subtraído do conteúdo de HL. Se até aqui Willie não tiver obtido uma recompensa, a instrução **jr nc** segue para a instrução **ret** e retorna. Caso contrário, o valor 1 é carregado no acumulador e armazenado em 57336. Esta é a posição de memória que a rotina principal verifica para saber se o escore deve ser incrementado e se é preciso começar nova tela. Feito isso, o processador encontra **ret** e retorna.

UM PASSO À FRENTE

O par DE é carregado com o valor 3 e HL, com 1548. Depois, a rotina BEEP, no endereço 949, é chamada executando o efeito sonoro da caminhada.

Antes de seguir adiante, porém, Willie deve ver o que há na sua frente. Sua posição é colocada no par HL. Adicionando 22561 a esse valor, obtemos a cor do caractere que se encontra 33 pontos

adiante do conteúdo de 57332 e 57333. Como este aponta para a cabeça de Willie, 33 é um caractere abaixo e à direita, indicando a posição imediatamente à frente de seus pés. O valor dessa posição é carregado no acumulador pela instrução **ld a,(hl)**.

A cor do caractere à frente dos pés de Willie é comparada com a cor do mar — 15, branco sobre azul — da cobra — 43, magenta sobre ciano — e da pedra — 42, vermelho sobre ciano. Se a cor do mar, da cobra ou da pedra estiverem presentes, a instrução **jr z,mdy** faz o processador saltar para a rotina que elimina Willie. Mas, se adiante dos pés de Willie estiver o caractere de encosta, o processador salta para a rotina que deixa o personagem parado — é evidente que ele não pode ir em frente a não ser pulando.

Investigar o caractere que está adiante de Willie não é tudo. Devemos verificar também em qual caractere ele estará pisando depois de se mover. Para isso, carrega-se DE com 32, que é adicionado ao conteúdo de HL, o que move o apontador 32 caracteres à frente — ou seja, para a linha de baixo.

A cor nessa posição é carregada no acumulador pela instrução **ld a,(hl)** e o conteúdo do acumulador é comparado com a cor do mar, a de um espaço vazio e a de uma cobra. Se um desses valores estiver presente, a instrução **jr z,mdy** vai para a rotina da morte.

Se nada disso aconteceu e Willie ainda está vivo, HL é carregado com a posição original do personagem. A é carregado com 40 — a cor de Willie —, BC, com 57016 — o endereço inicial dos padrões para a figura com as pernas abertas — e o par DE, com 514 — Willie com as pernas abertas ocupando um bloco de dois por dois caracteres. Em seguida, a rotina de impressão de bloco em 58970 é chamada e imprime na tela Willie com as pernas abertas.

O par HL é incrementado e carregado de volta em 57332 e 57333. Atualizamos, assim, a posição de Willie que, na próxima vez, estará um caractere à direita.

AINDA PARADO

A rotina **mts** é diretamente chamada quando Willie encontra a encosta à sua frente e não pode prosseguir a não ser pulando. Mas o processador também aciona essa rotina quando ele foi impresso com as pernas abertas.

Um valor 0 é carregado no acumulador e no endereço 57334, informando ao processador que, na próxima vez, o personagem deve ser impresso com as pernas juntas — mesmo que seja no mesmo lugar, se ele não tiver se movido, ou uma posição à frente, se foi impresso com as pernas abertas e sua posição foi incrementada.

A seguir, o processador retorna.

O ÚLTIMO SUSPIRO

Se Willie se afogou no mar, foi picado por uma cobra, caiu num buraco ou foi atingido por uma pedra, a rotina **mdy** é chamada. Para a divulgação de sua morte, coloca-se 2 no endereço 57336. Em algum ponto do programa, esse endereço será checado e o funeral de Willie será providenciado.

Depois disso, o processador retorna da rotina. O último **ret** da listagem, precedido do rótulo **jmp**, será apagado pela próxima parte de *Avalanche*. Sua única função consiste em impedir que haja erro na falta desta.

T

O programa a seguir faz Willie andar e verifica se ele se defrontou com algum perigo — ou recompensa.

```
10   ORG 19902
20   MAN LDD 18249
30   ANDB #31
40   CMPB #30
50   BNE MANI
60   LDA #1
70   STA 18252
80   MANI LDA 18261
90   LBNE JUM
100   LDX 18249
110   LEAX 544,X
120   LDX,X
130   CMPX #$5555
140   LBEQ MDY
150   CMPX #$AAAA
160   LBEQ MDY
170   CMPX #$5FF5
180   LBEQ MDY
190   CLRB
200   CLR 18264
210   LDA #$BF
220   STA $FF02
230   LDA $FF00
240   STA 18262
250   LDA #$DF
260   STA $FF02
270   LDA $FF00
280   STA 18263
290   CMPA #$F7
300   BNEE MANA
310   LDB #1
320   LDA 18262
330   CMPA #$F7
340   BNE MAND
350   LDB #129
360   MAND STB 18261
370   LDA 18264
380   BNE MANC
390   LDX 18249
400   PSHS X
410   BRA MMI
420   MANA LDA 18262
430   CMPA #$F7
440   BNE MAND
450   LDA #1
460   STA 18264
470   BRA MAND
480   MANC LDX 18249
490   LDU #1536
500   JSR CHARPR
510   LEAX 254,X
520   JSR CHARPR
530   LDX 18249
540   LEAX 1,X
550   PSHS X
560   LEAX 353,X
570   LDA ,X
580   CMPA #$D5
590   BEQ MDYA
600   CMPA #$FF
610   BEQ MDYA
620   CMPA #$50
630   BEQ MTS
640   LDA 18251
650   BEQ MMO
660   MMI LDX,S
670   LDU #17774
680   JSR CHARPR
690   LDX ,S
700   LEAX 256,X
710   JSR CHARPR
720   CLR 18251
730   BRA MANE
740   MMO LDX ,S
750   LDU #17814
760   JSR CHARPR
770   LDX ,S
780   LEAX 256,X
790   LDU #17846
800   JSR CHARPR
810   LDA #1
820   STA 18251
830   MANE LDX ,S
840   STX 18249
850   PULS X
860   RTS
870   MDYA PULS X
880   MDY LDA #2
890   STA 18252
900   RTS
910   MTS PULS X
920   LEAX -1,X
930   PSHS X
940   BRA MMI
950   JUM RTS
960   CHARPR EQU 19402
```

Essa rotina chama outras que ainda não estão na memória. Para evitar erros, não tente executá-la antes de ter colocado RTS nos lugares adequados.

RECOMPENSAS

A primeira parte da rotina procura saber se Willie alcançou suas recompensas. Para isso, a posição do personagem na tela é carregada no registrador D. Como existem 32 posições — 32 é 2↑5

—, precisamos investigar apenas os cinco últimos bits da posição de tela para trabalhar com a coordenada X de Willie. Se esta coordenada X tiver chegado a 30, a coordenada X do prêmio (cuja posição Y está fixa junto à encosta), Willie deve agarrá-lo.

A operação AND é feita, então, entre o conteúdo de B e 31 — B é o registrador menos significativo do par de registradores AB que compõe D. Isso isola os cinco bits menos significativos, que são comparados ao número 30.

Se os cinco últimos bits contêm 30, o valor 1 é carregado no acumulador e armazenado na variável morte, em 18252. Assim, o processador é informado de que deve trazer a próxima tela. Caso contrário, a instrução **BNE** faz o processador saltar essas instruções.

PULO OU PERIGO

A é carregado com o conteúdo de 18261, a variável do salto. Se não for 0, Willie estará saltando e o processador vai para a rotina **JUM.** Isso é feito com um desvio longo porque o rótulo está muito distante.

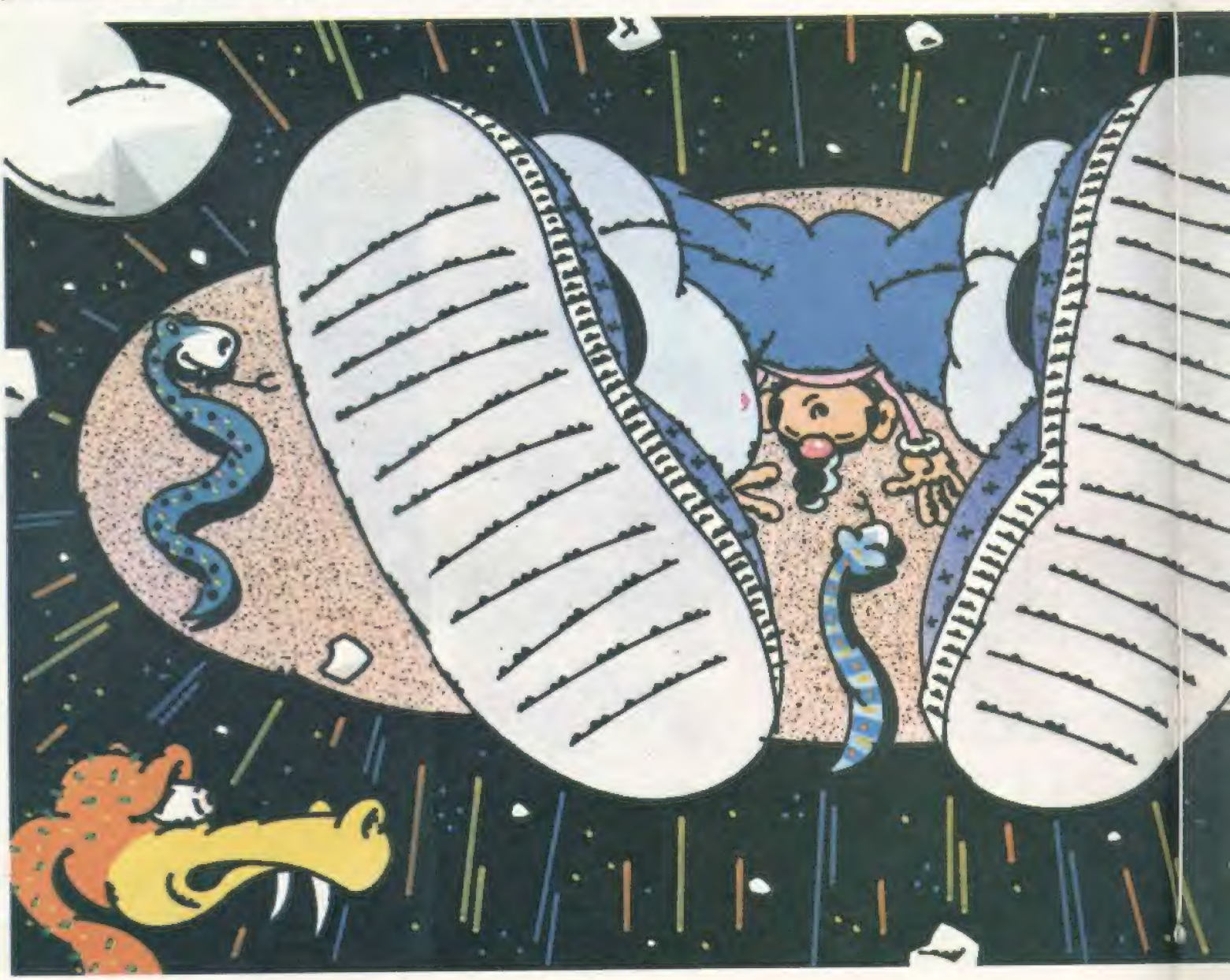
A tarefa seguinte é verificar se o personagem está diante de algum perigo fatal — um buraco, uma cobra ou o mar. A posição de Willie vai, então, de 18249 para o registrador X.

A instrução **LEAX 544,X** adiciona 544 à posição de Willie. Como ele tem dois caracteres de altura, precisamos acrescentar 2 × 256 (512) para apontar seus pés. Adicionamos 32 ao resultado, obtendo 544, porque há uma linha vazia de pontos entre os pés de Willie e o chão. O registrador X é carregado com o conteúdo dos dois bytes que estão sob os pés de Willie. Esse valor é comparado com $5555, $AAAA e $5FF5.

$5555 é o código de amarelo, que representa o céu — o que indicaria haver um buraco sob os pés de Willie; $AAAA é azul, a cor do mar, e $5FF5, vermelho sobre amarelo, a cor da cobra. Se uma dessas cores for encontrada sob seus pés, Willie está morto e a instrução **LBEQ** faz o processador saltar para a rotina **MDY**, que o elimina.

AS TECLAS

A tarefa seguinte consiste em verificar se alguma tecla foi pressionada. As



teclas M e N controlam, respectivamente, a corrida e o salto de Willie. Portanto, qualquer pressão sobre elas precisa ser detectada.

Primeiro limpa-se o registrador B, que será usado para carregar a variável que controla o salto de Willie. A posição de memória 18264 é ajustada com 0, já que irá armazenar a variável que controla a caminhada de Willie.

Para saber se uma tecla foi pressionada, deve-se analisar a matriz do teclado. No nosso caso, estamos interessados particularmente nas teclas M e N. A letra M fica na quarta coluna da sexta linha e a letra N, na quarta coluna da sétima linha.

Para examinar a situação de uma tecla, precisamos escrever o número da linha em $FF02; sua situação atual será dada em $FF00. Assim, para verificar a tecla N, armazena-se $BF em $FF02. $BF é 10111111 em binário, e é armazenado porque seleciona a sétima linha. O resultado em $FF00 é carregado no acumulador e armazenado em 18262.

M é examinada da mesma maneira, mas, desta vez, $DF — 11011111 em binário — é escrito em $FF02 e o resultado é armazenado em 18263. A instrução **CMPA #$F7** compara o resultado com $F7 ou 11110111. Lembre-se de que você está tentando ver se uma tecla da quarta linha foi pressionada — esta linha estará em zero. Se $F7 não está presente — e Willie não está saltando —, a instrução **BNE** desvia o processador para **MANA**. Mas, se $F7 estiver presente, B é carregado com 1 para indicar um salto vertical.

O resultado do exame da linha da tecla N, armazenado em 18262, é então carregado no acumulador e comparado com $F7. Observe que a tecla N está na mesma coluna que a tecla M, embora em linha diferente. Se o valor não for encontrado, **BNE** salta a instrução seguinte. Caso contrário, B é carregado com 129 para indicar que Willie dará um salto à frente. Quer seja 1, quer seja 129, o valor vai para 18261.

WILLIE ESTÁ ANDANDO?

O conteúdo da posição de memória 18264 é carregado no acumulador. Esta é a posição que armazena a variável que indica se Willie está andando ou não. Ela foi limpa, ou seja, ajustada com 0, no início do programa. Mas, neste ponto, não está necessariamente vazia porque a rotina que verifica se Willie está andando (que inicia no rótulo **MANA**) salta de volta para **MAND.**

Se o conteúdo da posição de memória 18264 não for 0 e Willie estiver andando, a instrução **BNE** faz o processador saltar para **MANC**, onde o programa começa a fazer Willie se mover. Caso contrário, o registrador X é carregado com o conteúdo de 18249 (endereço que armazena a posição de Willie na tela) e guardado na pilha. A instrução **BRA MNI** faz o processador saltar para **MNI**, que imprime o personagem na tela na posição definida pelo último valor armazenado na pilha.

Se M não foi pressionada e Willie não está pulando, o processador salta para **MANA**, para ver se ele está andando. O resultado do exame da tecla N, armazenado em 18262, é carregado de volta no acumulador e comparado com $F7. Se o valor não está presente, a instrução **BNE** manda o processador de volta para **MAND.** Como a posição de memória 18264 ainda está vazia, Willie é impresso no mesmo lugar. Isso significa que ele está efetivamente parado.

Mas, se N foi pressionada e Willie deve andar, 1 é carregado no acumulador e armazenado em 18264. Quando o processador saltar de volta para **MAND**, o personagem começará a andar.

APAGANDO A FIGURA ANTERIOR

Quando Willie começa a andar, é preciso apagá-lo de sua velha posição na tela. Do contrário, partes da figura serão deixadas pelo caminho.

X é carregado com a posição do personagem em 18249. O apontador de dados, U, é carregado com 1536, que corresponde ao canto superior esquerdo da tela. Assim, quando o processador saltar para a sub-rotina **CHARPR**, imprimirá dois caracteres de céu sobre Willie, apagando a metade superior da figura.

Para apagar a metade inferior de Willie, adiciona-se 254 a X e a rotina **CHARPR** é chamada outra vez. Soma-se 1 à posição da figura, que é carregada no registrador X — o que faz Willie se mover um caractere à frente. O resultado é armazenado na pilha.

Depois que Willie anda, precisamos verificar onde ele está pisando. A instrução **LEAX 353,X** incrementa o apontador que passa a indicar um byte imediatamente à frente das pernas do personagem, e **LDA ,X** carrega esse byte no acumulador. Ele é então comparado com $D5, a cor gráfica para a língua da cobra, e com $FF, a cor da pedra. Se algum desses valores estiver presente, Willie está condenado. A instrução **BEQ** manda então o processador para a rotina **MDYA**, que o elimina.

Esse byte também é comparado com $50, a cor gráfica correspondente à encosta da montanha. Se seu valor estiver presente, o processador salta para **MTS**, que volta a imprimir Willie no mesmo lugar, evitando que ele ande.

O QUE IMPRIMIR

A posição de memória 18251 é usada como uma baliza para informar ao processador qual das duas figuras será impressa na próxima vez — a que tem as pernas abertas ou a que tem as pernas juntas. Quando elas são impressas alternadamente, dão a impressão de que Willie está andando.

A baliza em 18251 é carregada no acumulador. Se está ajustada com 0, a instrução **BEQ** faz o processador pas-

sar para a rotina que começa com o rótulo **MNO**. Esta imprime a figura com as pernas abertas. Mas, se o conteúdo de 18251 não for 0 e a baliza estiver ajustada com 1, o salto não ocorre e o processador continua na rotina **MNI**, que imprime uma figura com as pernas juntas. Repare que, se Willie não estiver andando, o processador também pula para **MNI** e o imprime com as pernas juntas.

X é carregado com o último valor armazenado na pilha da máquina. Revendo a listagem, você constatará que o último dado colocado na pilha de máquina foi a nova posição de Willie, um caractere adiante de sua última posição. O apontador da pilha do usuário, U, que funciona como apontador de dados na rotina **CHARPR**, é carregado com 17774. Este é o endereço inicial dos dados da figura com as pernas juntas. O processador vai para a rotina **CHARPR** e imprime a metade superior de Willie.

O registrador X é carregado de novo e incrementado com 256, o que faz o apontador se mover um caractere para baixo — a posição da metade inferior de Willie. Observe que não foi preciso carregar U de novo. Ele é o apontador da pilha do usuário, e se ajusta automaticamente quando os dados são impressos na tela.

A posição de memória 18251 é ajustada com 0, para que a outra figura de Willie — a de pernas juntas — seja impressa na próxima vez. Em seguida, o processador vai para **MANE**.

Logo abaixo encontra-se a rotina **MMO**, que imprime a figura seguinte de Willie. Ela trabalha exatamente do mesmo modo. Desta vez, porém, U é ajustado com 17814, já que corresponde ao início dos dados para a figura com as pernas abertas. No final da rotina, 1 é carregado em A e armazenado em 18251, para que Willie seja impresso com as pernas juntas na próxima vez que a rotina de movimentação for chamada.

Qualquer que seja a figura impressa, o processador aciona a rotina **MANE**. Ela carrega a nova posição de nosso personagem em X e armazena-a em 18249. O último valor da pilha é colocado em X, para evitar que a pilha de máquina aumente descontroladamente. O processador então retorna.

MORTE OU IMOBILIDADE

Se Willie tiver sucumbido a algum dos perigos que o ameaçavam e estiver morto, o processador é mandado para **MDYA** ou **MDY** — o rótulo dependerá do local onde ele morreu. Se encontrou um perigo no início do programa, antes que a nova posição tivesse sido colocada na pilha, o processador irá para **MDY**. Mas, se a nova posição de Willie já foi colocada na pilha antes do passo fatal, ele irá para **MDYA**, e a nova posição será recuperada da pilha.

A é carregado com 2 — valor que indica a morte de Willie. Em seguida, esse valor é armazenado em 18252, posição da variável da morte.

Feito isso, o processador retorna.

Se Willie tiver esbarrado na encosta, o processador é mandado para **MTS**. Aqui, a nova posição de Willie é novamente recuperada da pilha, mas seu valor é utilizado e, depois de decrementado em 1, volta à pilha.

O processador salta mais uma vez para **MNI** e imprime Willie com as pernas juntas, na mesma posição de antes, o que o deixa efetivamente parado.

IX

O programa a seguir permite que Willie inicie sua corrida para o topo da montanha e verifica se ele encontrou algum perigo ou recompensa. Caso você esteja usando o Assembler de INPUT, precisará fazer uma alteração: na linha 5010, o comando **DIM T$(100)** muda para **DIM T$(200)**, uma vez que a rotina aqui apresentada é muito extensa.

```
10    org 54603
20    ld a,(-5202)
30    cp 0
40    jp nz,pl
50    ld a,(-5203)
60    cp 1
70    jp z,wa
80    ld hl,(62407)
90    ld de,(-5205)
100    add hl,de
110    push hl
120    dec hl
130    ld a,255
140    push hl
150    call 77
160    pop hl
170    ld de,32
180    add hl,de
190    ld a,255
200    call 77
210    pop hl
220    ld a,0
230    push hl
240    call 77
250    pop hl
260    ld de,32
270    add hl,de
280    ld a,1
290    push hl
300    call 77
310    pop hl
320    ld de,32
330    add hl,de
340    call 74
350    cp 72
360    jp z,mre
370    cp 74
380    jp z,mre
390    cp 37
400    jp z,mre
410    cp 254
420    jp z,mre
430    ld a,4
440    call 321
450    bit 2,a
460    jr nz,mf
470    ld b,1
480    bit 3,a
490    jr nz,ml
500    ld b,129
510    ml ld a,b
520    ld (-5202),a
530    jr mc
540    mf bit 3,a
550    jr nz,mc
560    ld a,1
570    ld (-5203),a
580    mc ld hl,(-5205)
590    ld de,222
600    sbc hl,de
610    jr nz,mo
620    ld a,1
630    ld (-5201),a
640    mo ret
650    wa ld hl,(62407)
660    ld de,(-5205)
670    add hl,de
680    ld de,33
690    add hl,de
700    push hl
710    call 74
720    pop hl
730    cp 36
740    jr z,mre
750    cp 52
760    jr z,mt
770    cp 13
780    jr z,mre
790    cp 17
800    jr z,mre
810    ld de,32
820    add hl,de
830    call 74
840    cp 72
850    jr z,mre
860    cp 74
870    jr z,mre
880    cp 254
890    jr z,mre
900    cp 37
910    jr z,mre
920    ld hl,(62407)
930    ld de,(-5205)
940    add hl,de
950    ld a,4
960    push hl
970    call 77
980    pop hl
990    inc hl
1000    ld a,6
1010    push hl
1020    call 77
1030    pop hl
1040    ld de,32
1050    add hl,de
1060    ld a,7
1070    push hl
```

```
1080    call 77
1090    pop hl
1100    dec hl
1110    ld a,5
1120    call 77
1130    ld hl,(-5205)
1140    inc hl
1150    ld (-5205),hl
1160    mt ld a,0
1170    ld (-5203),a
1180    ret
1190    mre ld a,2
1200    ld (-5201),a
1210    ret
1220    pl ret
1230    end
```

Você pode executar essa rotina mesmo sem as duas restantes, que completam o programa de movimentação de Willie. Embora esta parte possa chamar as demais, não haverá erro. Uma instrução **ret**, que apagaremos mais tarde, foi colocada no fim da rotina — assim, se uma seção inexistente for acessada, o programa simplesmente retornará.

O QUE WILLIE VAI FAZER?

A variável armazenada em −5202 indica se Willie vai ou não pular. O conteúdo dessa posição é carregado no acumulador e comparado com 0. Se não for 0, o personagem tem que pular e o processador salta para a rotina **pl**. Esta ainda não foi publicada e, no seu lugar, o processador encontra apenas uma instrução **ret**, voltando para a rotina principal do jogo.

Se Willie não deve pular, ou seja, se o conteúdo de −5202 é 0, o processador continua a execução da rotina. O conteúdo da posição de memória −5203, que guarda a variável da caminhada, é carregado no acumulador.

Há duas figuras para Willie — uma com as pernas juntas e a outra com as pernas abertas. Quando ele está parado, a primeira figura é impressa continuamente no mesmo lugar. Mas, quando está andando, as duas figuras são impressas alternadamente. A variável da caminhada indica ao processador se Willie deve andar uma posição — nesse caso, usa-se a figura de pernas abertas.

A instrução **cp 1** verifica se o personagem deve andar. Se o valor for 1, **jp z,wa** manda o processador para a rotina que faz Willie andar uma posição. Do contrário, o processador continua a executar a mesma rotina.

ANIMAÇÃO

Quando está andando, Willie abre as pernas e dá um passo à frente, aparecendo com as pernas juntas. Temos esta seqüência (cada item significa uma chamada na rotina de movimentação):

- Willie é impresso com as pernas juntas porque está imóvel; a tecla N foi pressionada e a variável, ajustada.
- Willie é impresso com as pernas abertas pela rotina que o faz andar; sua posição é incrementada e a variável da caminhada é ajustada com zero no fim da rotina.
- Willie é impresso com as pernas juntas, uma posição à frente, e as teclas são testadas.

Com as pernas abertas, o personagem ocupa duas posições adjacentes; com as pernas juntas, uma. Garantimos assim um efeito de movimentação suave.

Como você pode concluir da seqüência acima, toda vez que Willie for impresso com as pernas fechadas, a posição anterior deve ser apagada. Caso contrário, partes da figura serão deixadas pelo caminho.

O par HL é carregado com o endereço inicial da Tabela de Nomes (TN) da VRAM, que se encontra nas posições 62407 e 62408. A posição de Willie na tela, que está em −5205 e −5204, é colocada em DE. Os dois valores são somados em HL, que passa a conter o endereço de Willie na TN da VRAM. Esse endereço é guardado na pilha.

O endereço em HL é decrementado — o que corresponde a voltar uma posição na tela. Esse valor também é guardado na pilha. O acumulador é carregado com 255, o código do padrão vazio usado como céu, e a rotina 77 da ROM é chamada. Como você deve se lembrar, ela escreve o valor do acumulador no endereço da VRAM apontado por HL.

O último valor de HL é recuperado da pilha e adicionado a 32, por DE. Isso faz HL apontar para a posição logo abaixo da anterior. O acumulador é novamente carregado com 255, o padrão de céu, e a rotina 77 é chamada.

Todo esse procedimento teve como finalidade apagar a metade esquerda do nosso personagem que, com as pernas abertas, ocupa quatro posições. A metade direita será apagada quando a figura for impressa com as pernas juntas, sobreposta à anterior. Repare que isso poderia ser desnecessário se Willie já estivesse com as pernas juntas — mas é uma precaução sempre tomada.

IMOBILIDADE

Precisamos agora imprimir a nova figura com as pernas juntas. A posição que foi armazenada na pilha é recuperada em HL. O acumulador é carregado com 0, código do primeiro padrão para essa figura. Guarda-se o valor de HL na pilha e a rotina 77 é chamada.

Ao endereço, recuperado da pilha e colocado em HL, adiciona-se 32 — o que significa apontar uma posição abaixo. O acumulador é carregado com 1, código do segundo padrão da figura. O valor de HL é guardado na pilha e a rotina 77 é chamada. Feito isso, a figura de Willie com as pernas juntas já está impressa.

ONDE WILLIE ESTÁ PISANDO?

Como Willie se moveu uma posição à frente, convém verificar onde ele está pisando. Note que, mesmo que o personagem estivesse parado, esta parte da rotina seria executada. Se ele estiver pisando numa cobra, num buraco ou na água, o processador deve ser informado de sua morte.

A primeira coisa a fazer é verificar qual o código do padrão que está sob os pés de Willie. O endereço que o padrão dos pés ocupa na TN da VRAM é recuperado da pilha e colocado no par HL. O valor 32 é somado a esse endereço por meio do par DE, fazendo com que ele passe a apontar para a posição imediatamente abaixo dos pés de Willie. A rotina 74 da ROM é chamada e faz a leitura da VRAM, devolvendo a A o valor da posição apontada por HL.

Com isso, o acumulador passa a conter o código do padrão no qual Willie está pisando. Esse valor é comparado com 72 e 74, os padrões de mar; com 37, o padrão da cabeça da cobra, e com 254, o padrão do buraco. Se qualquer um desses valores estiver presente, a instrução **jp z,mre** manda o processador para a rotina **mre**, que elimina nosso personagem. Caso contrário, o processador continua executando a rotina — Willie está salvo.

HORA DE SALTAR

A impressão de Willie com as pernas juntas corresponde, efetivamente, ao fim da ação nesta parte da rotina. Desde que o personagem não tenha morrido, deveremos verificar se ele irá se mover na próxima vez. Para isso, examinamos a matriz do teclado.

O teclado do MSX é controlado pela interface de periféricos PPI (8255). Cada tecla faz parte de uma matriz e ocupa uma linha e coluna determinadas. Para verificar se uma tecla está sendo pressionada, mandamos um sinal para

a linha que ela ocupa e checamos se o bit correspondente à sua coluna tem valor 0. Essas operações envolvem escrita e leitura na PPI e podem ser implementadas com instruções **in** e **out** nos endereços adequados. No nosso caso, a rotina 321 da ROM realiza a tarefa. Para isso, basta colocar no acumulador a linha da matriz do teclado que queremos pesquisar. A rotina 321 devolve nos bits do acumulador a situação de todas as teclas dessa linha; se o bit correspondente à tecla for 0, ela está sendo pressionada.

Como as teclas M e N ocupam a quarta linha da matriz do teclado, o acumulador é carregado com 4. Em seguida, a rotina 321 é chamada. A instrução **bit 2,a** analisa o bit dois do número encontrado para verificar se a tecla M está sendo pressionada. Todos os bits correspondentes às teclas têm, normalmente, valor 1. Quando a tecla é pressionada, porém, eles assumem o valor 0. Se a tecla M foi pressionada, o bit dois tem valor 0.

Nesse caso, o processador ignora a instrução **jr nz**. Mas, se M não tiver sido pressionada, ele salta para a rotina **mf**. Como a tecla M foi pressionada, Willie deve pular; o registro B é carregado com 1.

Depois o bit três é testado da mesma maneira. Se N não foi pressionada, o bit três tem valor 1 e a instrução **jr nz** faz o processador saltar a próxima instrução. Mas, se a tecla N foi pressionada, **jr nz** não tem efeito e B é carregado com 129.

Seja como for, o processador transfere o conteúdo de B para A e o coloca em -5202. Esta, como você se lembra, é a posição de memória que o processador examina no início da rotina, para ver se Willie deve ou não pular.

O significado dos números 1 e 129 será explicado nas duas próximas seções de *Avalanche*, que executam o salto vertical e o salto à frente.

A instrução **jr mc** faz o processador pular a próxima parte.

WILLIE DEVE ANDAR?

Se a tecla M não foi pressionada, o processador vai para a rotina **mf**, que checa se a tecla N foi pressionada — ou seja, se Willie não irá pular, mas, simplesmente, andar.

O bit três é testado para ver se a tecla N foi pressionada. Note que a leitura da matriz do teclado ainda se encontra no acumulador — o processador pulou para esta parte da rotina depois que o bit dois foi testado.

Se a tecla N não tiver sido pressionada, a instrução **jr nz** salta a próxima parte. Caso contrário, A é carregado com 1 e esse valor vai para o endereço -5203, onde se encontra a variável da caminhada, examinada no início da rotina. Nessa posição o valor 1 indica que deve ser impressa a figura de Willie que tem as pernas abertas — portanto, ele está andando.

PRÊMIOS

Falta verificar apenas um elemento: Willie pegou o seu prêmio?

A posição do personagem é armazenada nos endereços -5205 e -5204. O conteúdo desses endereços é dado pelo par HL. O valor 222, que corresponde à posição de tela onde o prêmio foi impresso, é colocado no par DE.

O conteúdo de DE é subtraído do conteúdo de HL. Se o resultado for diferente de 0, Willie não alcançou o prêmio. A instrução **jr nz,mo** salta então as próximas instruções. Mas, se o resultado for 0, o personagem já obteve prêmio e o valor 1 é carregado em -5201 através do acumulador. Esta é a posição que a rotina principal examina, informando se o escore deve ser aumentado e outro nível iniciado. Feito isso, o processador encontra a instrução **ret** e retorna.

UM PASSO À FRENTE

Antes de prosseguir, Willie deve verificar o que existe à sua frente. Sua posição é colocada no par DE; o endereço inicial da TN da VRAM, em HL. Os dois valores são somados em HL, que passa a conter o endereço na TN que corresponde à posição da cabeça de Willie na tela. O valor 33 é somado em HL, movendo o apontador uma linha para baixo e uma posição para a direita — em outras palavras, para a posição adiante dos pés de Willie. Esse valor é guardado na pilha e, em seguida, a rotina 74 da ROM é chamada para fazer a leitura da VRAM.

O código do padrão que está na frente dos pés de Willie se encontra no acumulador. Ele é comparado com 36, padrão da língua da cobra; com 13 e 17, padrões da pedra, e com 52, padrão da encosta. Se os padrões da pedra ou da cobra estão presentes, a instrução **jr z,mre** faz o processador saltar para a rotina que elimina Willie. Mas, se na posição estiver o padrão da encosta, o processador salta para a rotina que deixa Willie parado. Obviamente, ele não pode ir em frente a não ser pulando.

Não basta, porém, identificar o padrão que está na frente de Willie. Precisamos saber, também, em que padrão ele pisará depois que se mover. Para isso, DE é carregado com 32 e somado ao conteúdo de HL, que aponta para uma posição abaixo da anterior.

A rotina 74 é novamente chamada e o conteúdo do acumulador comparado com os padrões do mar, do buraco e da língua da cobra. Se qualquer um desses valores estiver presente, a instrução **jr z,mre** vai para a rotina da morte. Se nada disso aconteceu e o personagem ainda está vivo, o par HL é carregado com o endereço inicial da TN, e DE, com a posição de Willie na tela. Esse valor é somado em HL.

A rotina 77 é chamada quatro vezes para imprimir os quatro padrões da figura de Willie com as pernas abertas, cujos códigos são 4, 5, 6 e 7.

A posição do personagem é carregada de -5205 e -5204 para o par HL, incrementada e devolvida. Com isso, atualizamos a posição de Willie. Na próxima vez, ele começará deslocado um caractere para a direita.

AINDA PARADO

A rotina **mt** é chamada diretamente quando Willie se depara com a encosta, não podendo prosseguir a não ser com um pulo. O processador, porém, chega a essa rotina quando o personagem foi impresso com as pernas abertas.

Um valor 0 é carregado no endereço -5203, por intermédio do acumulador, indicando ao processador que, na próxima vez, a figura de Willie deve ser impressa com as pernas juntas, no mesmo lugar ou uma posição à frente, dependendo do que aconteceu.

Em seguida, o processador retorna.

O ÚLTIMO SUSPIRO

Se Willie se afogou no mar, foi picado por uma cobra, caiu num buraco ou foi atingido por uma pedra, a rotina **mre** é chamada, informando ao resto do programa que Willie morreu.

Para que se divulgue essa informação, o valor 2 é colocado no endereço -5201, que será examinado em algum ponto do jogo. As providências para o funeral de Willie serão, então, tomadas.

Feito isso, o processador retorna.

O último **ret** da listagem, precedido do rótulo **pl**, será apagado pela próxima rotina de *Avalanche*. Sua única função é evitar que, na falta das rotinas, ocorra algum erro.

UMA PLANILHA ELETRÔNICA (3)

- COMO DAR ENTRADA ÀS EQUAÇÕES
- CÓPIA ABSOLUTA E RELATIVA
- O USO DE CONSTANTES
- COMPLETANDO O PROGRAMA

Com este artigo, encerramos nossa série sobre planilhas eletrônicas. Depois de acrescentar ao programa as linhas aqui apresentadas, você poderá explorar todo o seu potencial.

Para ter uma planilha completa e operacional, carregue na memória do seu micro as partes já dadas do programa e acrescente as linhas que faltam. Nos artigos anteriores, mostramos, por meio de exemplos, como montar uma planilha. Agora, que você já tem todo o programa, veremos como utilizá-lo.

As instruções que se seguem são válidas para todas as máquinas. As diferenças existentes entre o Spectrum e os demais microcomputadores serão sempre especificadas.

MONTAGEM DAS EQUAÇÕES

O que faz da planilha eletrônica uma ferramenta tão versátil é sua capacidade de trabalhar com equações. Não se assuste com a palavra equação: nossas "equações" não passam de especificações bastante simples das operações aritméticas de adicionar valores das colunas, obter o total de uma determinada linha, calcular porcentagem etc. Os ope-

MICRO DICAS

USE A PLANILHA COM MAIS EFICIÊNCIA

Agora que você dispõe de um programa para a elaboração de uma planilha de cálculo, convém conhecer alguns "truques" que lhe permitirão utilizá-la de maneira bem mais rápida e eficiente:

• Preenchimento da planilha: depois de definir o título geral e os subtítulos das seções, dedique-se ao "esqueleto" da planilha, ou seja, à identificação das linhas e colunas. Em seguida, coloque os dados propriamente ditos e, finalmente, as fórmulas. Procedendo assim, você diminuirá as chances de cometer erros de entrada e evitará que se executem cálculos antes que todos os dados e rótulos estejam presentes.

• Cuidado com os laços sem fim: esse tipo de problema pode ocorrer quando se colocam duas fórmulas em células diferentes, uma utilizando o resultado do cálculo da outra. Em conseqüência, o programa "emperra", pois a detecção de uma situação como esta não foi prevista. Não deixe de verificar todas as fórmulas antes de executar o programa.

radores usados são: mais (+), menos (−), vezes (*), dividido (/), por cento (%), total de linhas (&) e total de colunas ($).

Para escrever uma equação, indica-se o nome da primeira célula, depois o nome da segunda célula e, por último, o operador. Vejamos alguns exemplos. Se colocarmos A1A2+ na célula C1, obteremos a soma dos valores contidos em A1 e A2 na célula C1. A equação A1A10$ soma todos os valores da coluna A, da linha 1 até a linha 10. Da mesma maneira, A6F6& soma todos os valores da linha 6, começando na coluna A e terminando na coluna F. Finalmente, A5B2% é calculado como A5*B2/100, o que dá B2 por cento de A5. As equações são colocadas nas células em que se quer que apareçam os resultados.

Com exceção das versões destinadas ao Spectrum e ao Apple, o programa permite ainda que se acrescente um número no fim da equação para especificar a quantidade de casas decimais que aparecem na resposta.

CÓPIA ABSOLUTA E RELATIVA

O programa oferece a alternativa de se copiar o conteúdo de uma célula. Isso nos poupa o trabalho de redigitar uma fórmula em todas as células em que será usada. Podemos escolher entre uma cópia absoluta ou relativa. A cópia absoluta produz uma réplica idêntica da célula, onde quer que você queira, seja numa única célula, seja numa coluna inteira. A cópia relativa, usada especialmente para equações, sofre alterações conforme a linha e a coluna para onde está sendo copiada.

Suponhamos que você tem A1B1* na célula C1 e quer que toda célula da coluna C contenha o produto das duas colunas anteriores — ou seja, A2B2* em C2, A3B3* em C3 etc. A cópia relativa faz isso para você. O procedimento é o seguinte: pressione a tecla de cópia (verifique, adiante, qual a tecla que seu micro usa), selecione R (para relativa) e coloque o cursor sobre a célula a ser copiada (no Spectrum, digita-se o nome da célula). Em seguida, defina se a cópia deve ser feita em linha ou coluna — no nosso caso, pressione C. Finalmente, digite o nome da célula inicial (C1) e da final (C10, por exemplo). O programa, assim, preencherá toda a coluna para você.

Experimente também copiar valores em linhas e obter cópias absolutas de equações e valores.

CONSTANTES

Precisamos, com freqüência, utilizar constantes em uma equação — por exemplo, um valor como 17, para calcular a porcentagem do ICM, ou como 10, para calcular descontos.

Como não se pode colocar números diretamente em uma equação — que é baseada no conteúdo de duas células —, empregam-se diferentes métodos para a introdução de constantes.

No Spectrum, elas costumam ser precedidas pela letra Z. Assim, A1Z17% calcula o ICM de um valor colocado na célula A1. Nos demais micros, as constantes devem ser colocadas na coluna Z. Retomando o último exemplo: o valor 17 será colocado na célula Z1 e a equação tomará a forma A1Z1%.

Esse procedimento é necessário porque os valores das constantes não devem ser alterados nem mesmo quando se opta por uma cópia relativa. Se reservarmos uma coluna para elas, estaremos assegurando um tratamento diferenciado na hora da cópia.

O USO DO PROGRAMA

Existem algumas variações, de máquina para máquina, no que diz respeito à colocação de valores, rótulos e equações na planilha, assim como à execução das operações. Comentaremos, em seguida, os detalhes específicos para cada microcomputador.

S

O programa do Spectrum demora um pouco para definir as variáveis internas. Assim, você terá uma tela vazia por alguns momentos após teclar **RUN**. A tela aparece no modo de valores — o que significa que valores e rótulos podem ser inseridos. Pressione E para passar ao modo de equação e V para voltar ao de valores. Para introduzir um dado em uma célula, use as teclas de controle do cursor; ao chegar à célula correta, pressione I. Em seguida, digite o que quiser.

As equações são mostradas sobre fundo amarelo, com os valores em preto e os rótulos em azul. Considera-se rótulo qualquer dado digitado no modo de valores que comece com uma letra. Como as cores aparecem em qualquer modo, você pode distinguir facilmente as células que contêm equações mesmo que esteja vendo a folha no modo de valores ou vice-versa.

A tela exibe apenas quatro colunas e dez linhas de cada vez, mas é possível mudar para outra parte da planilha pressionando-se simultaneamente as teclas de controle do cursor e <**SYMBOL SHIFT**>.

Tente inserir alguns valores e rótulos, mude para o modo de equações e some os valores das colunas e linhas. Para ver o resultado dos cálculos, volte para a tela de valores e pressione C e <**SYMBOL SHIFT**>. Se algum número for muito grande e não couber na célula, os dígitos da direita serão eliminados e os que permanecerem na tela ficarão piscando como aviso.

Agora, experimente copiar algumas equações e valores usando a tecla Z. O programa pedirá toda a informação que for necessária.

Segue-se a lista de comandos usados pelo programa:

V mostra a tela de valores;
E mostra a tela de equações;
I insere um dado em uma célula;
I seguido de <**ENTER**> apaga um valor ou rótulo;
I seguido de # apaga uma equação;
C mais <**SYMBOL SHIFT**> calcula os valores da planilha;
<**SPACE**> mais <**SYMBOL SHIFT**>

interrompe os cálculos;
Z copia uma célula;
S mais <**SYMBOL SHIFT**> grava dados em fita;
J mais <**SYMBOL SHIFT**> carrega dados da fita;
P imprime a tela em uma impressora;
Teclas de controle do cursor movem o cursor;
Teclas de controle mais <**SYMBOL SHIFT**> movem a janela de texto.

TRS-Color

Ao executar o programa, você verá apenas a parte superior esquerda da planilha; se quiser examinar qualquer outra parte, recorra às teclas de controle do cursor (no micro Apple, mova o cursor para cima e para baixo usando as teclas A e Z). Você pode também passar diretamente para qualquer ponto da planilha digitando G.

As teclas V e E alternam as telas de equações e valores. É possível entrar valores nas duas telas, mas os resultados só serão mostrados na tela de valores. Para inserir um dado em uma célula, mova o cursor até ela e pressione I. Depois, digite sua entrada.

As equações são digitadas conforme descrevemos anteriormente. No TRS-Color e no MSX podemos discriminar o número de casas decimais do resultado. A1B1/2, por exemplo, mostra o resultado de A1 dividido por B1 com duas casas decimais. Esse recurso é muito útil quando estamos lidando com valores financeiros. No caso de obtermos números que excedam o tamanho da célula, a mensagem <OV> será exibida.

Eis as teclas usadas no programa:
V mostra a tela de valores;
E mostra a tela de equações;
I insere um rótulo, valor ou equação;
C copia uma célula;
S grava dados no cassete ou disco;
L lê dados do cassete ou disco;
Q sai do programa;
Teclas de controle do cursor movem o cursor e a janela de texto (no Apple, usam-se A e Z para mover o cursor para cima e para baixo).

S

```
910 IF z=3 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2): LET v(3)=(CODE s$(3))-64
: LET v(4)=VAL s$(4 TO 5): LET
o$=s$(6): RETURN
920 IF z=4 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2 TO 3): LET v(3)=(CODE s$(4
))-64: LET v(4)=VAL s$(5 TO 6)
: LET o$=s$(7): RETURN
930 IF z=5 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2): LET v(3)=(CODE s$(3))-64
: LET v(4)=VAL s$(4): LET o$=s
$(5): RETURN
940 IF z=6 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2 TO 3): LET v(3)=(CODE s$(4
))-64: LET v(4)=VAL s$(5): LET
o$=s$(6): RETURN
950 IF z=7 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2): LET v(3)=(CODE s$(3))-64
: LET v(4)=VAL s$(4 TO 5): LET
o$=s$(6): RETURN
960 IF z=8 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2 TO 3): LET v(3)=(CODE s$(4
))-64: LET v(4)=VAL s$(5 TO 6)
: LET o$=s$(7): RETURN
970 IF z=9 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2): LET v(3)=(CODE s$(3))-64
: LET v(4)=VAL s$(4 TO 6): LET
o$=s$(7): RETURN
980 IF z=10 THEN  LET v(1)=(
CODE s$(1))-64: LET v(2)=VAL s
$(2 TO 3): LET v(3)=(CODE s$(4
))-64: LET v(4)=VAL s$(5 TO 7
): LET o$=s$(8): RETURN
990 LET v(1)=(CODE s$(1))-64:
LET v(2)=VAL s$(2): LET v(3)=(
CODE s$(3))-64: LET v(4)=VAL s
$(4 TO 7): LET o$=s$(8):
RETURN
1000 RETURN
1010 IF d$(v(2),v(1),1)>"9" THE
N  LET t=0: RETURN
1020 IF v(3)=26 THEN  LET b=v(4
): LET a=VAL d$(v(2),v(1), TO 8
): GOTO 1050
1030 IF d$(v(4),v(3),1)>"9" THE
N  LET t=0: RETURN
1040 LET a=VAL d$(v(2),v(1), TO
 8): LET b=VAL d$(v(4),v(3), TO
 8)
1050 IF o$="+" THEN  LET t=a+b
 RETURN
1060 IF o$="-" THEN  LET t=a-b
 RETURN
1070 IF o$="/" AND b=0 THEN  LI
T t=0: RETURN
1080 IF o$="/" THEN  LET t=a/b:
 RETURN
1090 IF o$="*" THEN  LET t=a*b:
 RETURN
1100 IF o$="↑" THEN  LET t=(a↑t
)/100: RETURN
1110 IF o$="$" THEN  LET t=0: F
OR s=v(2) TO v(4): LET t=t+VAL
d$(s,v(1), TO 8): NEXT s: RETUR
N
1120 IF o$="&" THEN  LET t=0: F
OR s=v(1) TO v(3): LET t=t+VAL
d$(v(2),s, TO 8): NEXT s: RETUR
N
1130 RETURN
1140 IF z=1 THEN  IF VAL a$(2)=
0 OR VAL a$(4)=0 THEN  LET f=1:
 RETURN
1150 IF z=2 THEN  IF VAL a$(2 T
O 3)>30 OR VAL a$(2 TO 3)=0 OR
VAL a$(5)=0 THEN  LET f=1: RETU
RN
1160 IF z=3 THEN  IF VAL a$(4 T
O 5)>30 OR VAL a$(4 TO 5)=0 OR
VAL a$(2)=0 THEN  LET f=1: RETU
RN
1170 IF z=4 THEN  IF VAL a$(2 T
O 3)>30 OR VAL a$(5 TO 6)>30 OR
 VAL a$(2 TO 3)=0 OR VAL a$(5 T
O 6)=0 THEN  LET f=1: RETURN
1180 IF z=5 OR z=7 OR z=11 THEN
  IF VAL a$(2)=0 THEN  LET f=1:
 RETURN
1190 IF z=6 OR z=8 OR z=10 THEN
  IF VAL a$(2 TO 3)=0 OR VAL a$
(2 TO 3)>30 THEN  LET f=1: RETU
RN
1200 LET s$="          ": FOR g=1
 TO c: LET s$(g)=a$(g): NEXT g:
 GOSUB 890: IF o$="&" AND (v(1)
>v(3) OR v(2)<>v(4)) THEN  LET
f=1: RETURN
1210 IF o$="$" AND (v(1)<>v(3)
OR v(2)>v(4)) THEN  LET f=1: RE
TURN
1220 LET f=0: RETURN
1230 LET g$=d$(y+wy,x+wx,17)
1240 PRINT  FLASH 0+(g$="5" AND
 t$="VAL"); INK 0+(2 AND g$="4"
); PAPER 7-(2 AND g$="2")-(g$="
1" OR g$="5");AT y*2+2,x*9+5;(d
$(y+wy,x+wx, TO 8) AND t$="VAL"
);(d$(y=wy,x+wx,9 TO 16) AND t$
="IGUA"): RETURN
1250 SOUND .4,10
1260 LET s$="          ": LET os=
0
1270 PAUSE 20: PRINT AT cy*2,(c
x-1)*9+5+os; BRIGHT 1; OVER 1;"
 "
1280 IF CODE INKEY$>30 AND CODE
 INKEY$<128 THEN  LET s$(os+1)=
INKEY$: PRINT AT cy*2,(cx-1)*9+
5+os;s$(os+1): LET os=os+1: GOT
O 1350
1290 IF CODE INKEY$=12 AND os>0
 THEN  LET os=os-1: LET s$(os+1
)="": PRINT AT cy*2,((cx-1)*9)+
5+os; OVER 0; BRIGHT 0;"  ": GO
TO 1270
1300 IF CODE INKEY$=13 AND s$="
          " THEN  RETURN
1310 IF CODE INKEY$=13 AND t$="
VAL" AND CODE s$(1)>64 THEN  LE
T d$(wy+cy-1,wx+cx-1,17)="2": L
ET d$(wy+cy-1,wx+cx-1, TO 8)=s$
: LET y=cy-1: LET x=cx-1: GOSUB
 1230: RETURN
1320 IF CODE INKEY$=13 AND t$="
VAL " THEN  GOSUB 1380: LET os=
0: LET d$(wy+cy-1,wx+cx-1, TO 8
)=s$: LET y=cy-1: LET x=cx-1: G
OSUB 1230: RETURN
1330 IF CODE INKEY$=13 AND t$="
IGUA" THEN  GOSUB 1380: LET os=
0: LET d$(wy+cy-1,wx+cx-1,9 TO
16)=s$: LET x=cx-1: LET y=cy-1:
 GOSUB 1230: RETURN
1340 GOTO 1270
1350 IF os=8 AND t$="VAL" THEN
 SOUND .1,-10: GOSUB 1380: LET
d$(wy+cy-1,wx+cx-1, TO 8)=s$: L
```

```
ET y=cy-1: LET x=cx-1: GOSUB 12
30: RETURN
1360 IF os=8 AND t$="IGUA" THEN
  SOUND .1,-10: GOSUB 1380: LET
 d$(wy+cy-1,wx+cx-1,9 TO 16)=s$
  LET y=cy-1: LET x=cx-1: GOSUB
 1230: RETURN
1370 GOTO 1270
1380 PRINT AT cy*2,((cx-1)*9)+5
+os; BRIGHT 0; OVER 1;" "
1390 IF t$="IGUA" THEN  GOSUB 1
490: RETURN
1400 GOSUB 1410: PRINT AT cy*2,
(cx-1)*9+5;s$: RETURN
1410 LET b$="          000": FOR
u=1 TO os: LET b$(u+(8-os))=s$(
u): NEXT u
1420 FOR u=1 TO 11
1430 IF b$(u)<>"." THEN  NEXT u
1440 IF u>=11 THEN  LET b$(9)="
.": LET s$=b$(4 TO ): RETURN
1450 LET w=6-u
1460 IF w<0 THEN  LET w=ABS w+1
: LET s$=b$(w TO w+8): RETURN
1470 LET s$="          "
1480 FOR u=1 TO 8-w: LET s$(w+u
)=b$(u): NEXT u: RETURN
1490 IF s$(1)="▌" THEN  LET s$=
"          ": LET d$(wy+cy-1,wx+c
x-1,17)="0": RETURN
1500 LET a$="": FOR z=1 TO 8: L
ET a$=a$+(s$(z) AND s$(z)<>" ")
1510 NEXT z: LET c=LEN a$
1520 IF c<4 THEN  LET d$(wy+cy-
1,wx+cx-1,17)="4": RETURN
1530 RESTORE 1630: FOR z=1 TO 1
1: LET f=0: READ m$
1540 FOR w=1 TO c
1550 IF m$(w)="A" THEN  GOSUB 1
650: IF f THEN  GOTO 1610
1560 IF m$(w)="N" THEN  GOSUB 1
670: IF f THEN  GOTO 1610
1570 IF m$(w)="O" THEN  GOSUB 1
690: IF f THEN  GOTO 1610
1580 IF m$(w)="Z" THEN  GOSUB 1
710: IF f THEN  GOTO 1610
1590 NEXT w: GOSUB 1140: IF NOT
 f THEN  LET s$="          ": FOR
 w=1 TO c: LET s$(w)=a$(w): NEX
T w: LET d$(wy+cy-1,wx=cx-1,18)
=CHR$ z: LET d$(wy+cy-1,wx+cx-1
,17)="1": RETURN
1600 GOTO 1620
1610 NEXT z
1620 LET d$(wy+cy-1,wx+cx-1,17)
="4": RETURN
1630 DATA "ANANOOO","ANNANOO","
ANANNOO","ANNANNO"
1640 DATA "ANZNOOOO","ANNZNOOO"
,"ANZNNOOO","ANNZNNOO","ANZNNNO
O","ANNZNNNO","ANZNNNNO"
1650 IF a$(w)>="A" AND a$(w)<="
X" THEN  RETURN
1660 LET f=1: RETURN
1670 IF a$(w)>="0" AND a$(w)<="
9" THEN  RETURN
1680 LET f=1: RETURN
1690 LET c$=a$(w): IF c$="+" OR
 c$="-" OR c$="*" OR c$="/" OR
c$="%" OR c$="$" OR c$="&" THEN
  RETURN
1700 LET f=1: RETURN
1710 IF a$(w)="Z" THEN  RETURN
1720 LET f=1: RETURN
1730 FOR y=1 TO 30: FOR x=1 TO
24: LET d$(y,x, TO 8)="     0.00
": LET d$(y,x,17)="0": LET d$(y
,x,18)=CHR$ 0: NEXT x: NEXT y
1740 RESTORE 1750: FOR x=USR "a
" TO USR "c"+7: READ a: POKE x,
a: NEXT x: RETURN
1750 DATA 8,8,8,8,8,8,8,8
1760 DATA 0,0,0,255,0,0,0,0
1770 DATA 8,8,8,255,8,8,8,8
1780 INPUT "NOME DO ARQUIVO ?";
n$
1790 SAVE n$ DATA d$()
1800 PRINT #1;AT 0,0;"PRESSIONE
 V PARA VERIFICAR "
1810 PAUSE 0: IF INKEY$="V" OR
INKEY$="v" THEN  GOTO 1830
1820 RETURN
1830 VERIFY n$ DATA d$(): RETUR
N
1840 INPUT "NOME DO ARQUIVO ?";
n$
1850 LOAD n$ DATA d$(): RETURN
```

```
1490 PRINT @448:PRINT @448,;:IF
 LEN(D$(CC,CR))=1 THEN PRINT"NA
DA A COPIAR":SOUND 5,5:FOR K=1
TO 500:NEXT:RETURN ELSE PRINT "
COPIA aBSOLUTA OU rELATIVA";
1500 A$=INKEY$:IF A$="" THEN 15
00
1510 IF A$=CHR$(13) THEN RETURN
1520 IF A$="A" THEN TC=1:GOTO 1
540
1530 IF A$<>"R" THEN 1490 ELSE
TC=0
1540 PRINT @448
1550 IF TC=0 AND ASC(D$(CC,CR))
<>131 THEN PRINT @448,"ERRADO :
 MODO COPIA - SOMENTE   EQUACOE
S";:SOUND 1,5:FOR D=1 TO 500:NE
XT D:GOTO 490
1560 PRINT @448:PRINT @448,"COP
IA DE lINHA OU cOLUNA";
1570 A$=INKEY$:IF A$="" THEN 15
70
1580 IF A$=CHR$(13) THEN RETURN
1590 IF A$="C" THEN DC=1:GOTO 1
610
1600 IF A$<>"L" THEN 1560 ELSE
DC=0
1610 PRINT @448:PRINT @448,"COM
ECAR PELA CELULA";:INPUT A$:IF
A$="" THEN RETURN
1620 S1=ASC(A$)-64:IF S1<1 OR S
1>25 THEN 1610
1630 S2=VAL(MID$(A$,2)):IF S2<1
 OR S2>30 THEN 1610
1640 PRINT @448:PRINT @448,"TER
MINAR NA CELULA";:INPUT A$
1650 IF A$="" THEN RETURN
1660 C1=ASC(A$)-64:IF C1<1 OR C
1>25 THEN 1640
1670 C2=VAL(MID$(A$,2)):IF C2<1
 OR C2>30 THEN 1640
1680 PRINT @448,"COPIANDO  -  E
SPERE"
```


Entrada das equações no Spectrum.

Cálculo dos resultados na tela do Spectrum.

```
1690 IF TC=1 THEN 1970
1700 IF DC=1 THEN 1840
1710 IF C2<>S2 THEN 2090
1720 AS=ASC(MID$(D$(CC,CR),2))-
64:IF AS<>26 AND AS+C1-S1>25 TH
EN C1=25+S1-AS
1730 AS=ASC(MID$(D$(CC,CR),5))-
64:IF AS<>26 AND AS+C1-S1>25 TH
EN C1=25+S1-AS
1740 D$(S1,S2)=D$(CC,CR)
1750 FOR I=S1+1 TO C1
1760 A$=D$(I-1,S2)
1770 IF MID$(A$,2,1)="Z" THEN 1
790
1780 MID$(A$,2,1)=CHR$(ASC(MID$
(A$,2,1))+1)
1790 IF MID$(A$,5,1)="Z" THEN 1
810
1800 MID$(A$,5,1)=CHR$(ASC(MID$
(A$,5,1))+1)
1810 D$(I,S2)=A$:NEXT
1820 IF C1>CX THEN CX=C1
1830 GOTO 2080
1840 IF C1<>S1 THEN 2090
1850 AS=VAL(MID$(D$(CC,CR),3,2)
):IF AS+C2-S2>30 THEN C2=30+S2-
AS
1860 AS=VAL(MID$(D$(CC,CR),6)).
IF AS+C2-S2>30 THEN C2=30+S2-AS
1870 D$(S1,S2)=D$(CC,CR)
1880 FOR I=S2+1 TO C2
1890 A$=D$(S1,I-1)
1900 IF MID$(A$,2,1)="Z" THEN 1
920
1910 V=VAL(MID$(A$,3,2))+1:IF V
<10 THEN MID$(A$,3,2)=STR$(V) E
LSE MID$(A$,3,2)=MID$(STR$(V),2
,2)
1920 IF MID$(A$,5,1)="Z" THEN 1
940
1930 V=VAL(MID$(A$,6,2))+1:IF V
<10 THEN MID$(A$,6,2)=STR$(V) E
LSE MID$(A$,6,2)=MID$(STR$(V),2
,2)
1940 D$(S1,I)=A$:NEXT
1950 IF C2>RX THEN RX=C2
1960 GOTO 2080
1970 IF DC=1 THEN 2030
1980 IF C2<>CR THEN 2090
1990 FOR I=CC+1 TO C1
2000 D$(I,CR)=D$(I-1,CR):NEXT
2010 IF C1>CX THEN CX=C1
2020 GOTO 2080
2030 IF C1<>CC THEN 2090
2040 FOR I=CR+1 TO C2
2050 D$(CC,I)=D$(CC,I-1):NEXT
2060 IF C2>RX THEN RX=C2
2070 GOTO 2080
2080 MO=1:GOSUB 70:RETURN
2090 PRINT @448,"erro NO DESTIN
O":SOUND 1,5:FOR D=1 TO 500:NEX
T D
2100 GOTO 2080
```

```
1490 LOCATE 0,21:PRINTSPC(35):L
OCATE 0,21:IF LEN(D$(CC,CR))=1
THEN PRINT"NADA PARA COPIAR":FO
R K=1 TO 500:NEXT:RETURN ELSE P
RINT"CóPIA <A>BSOLUTA OU <R>ELA
TIVA: ";:
1500 A$=INKEY$:IF A$="" THEN 15
00
1510 IF A$=CHR$(13) THEN RETURN
1520 IF A$="A" THEN TC=1:GOTO 1
540
1530 IF A$<>"R" THEN 1490 ELSE
TC=0
1540 LOCATE 0,21:PRINTSPC(35):L
OCATE 0,21
1550 IF TC=0 AND ASC(D$(CC,CR))
<>131 THEN PRINT"MODO DE CóPIA
ERRADO - SÓ EQUAÇÕES!";:FOR D=1
 TO 500:NEXT:GOTO 490
1560 LOCATE 0,21:PRINT"CóPIA PO
R <LINHA> OU <COLUNA>:    ";
1570 A$=INKEY$:IF A$="" THEN 15
70
1580 IF A$=CHR$(13) THEN RETURN
1590 IF A$="C" THEN DC=1:GOTO 1
610
1600 IF A$<>"L" THEN 1570 ELSE
DC=0
1610 LOCATE 0,21:PRINTSPC(37):L
OCATE 0,21:PRINT"INICIA NA CéLU
LA";:INPUT A$:IF A$="" THEN RET
URN
1620 S1=ASC(A$)-64:IF S1<1 OR S
1>25 THEN 1610
1630 S2=VAL(MID$(A$,2)):IF S2<1
 OR S2>30 THEN 1610
1640 LOCATE 0,21:PRINTSPC(37):L
OCATE 0,21:PRINT"TERMINA NA CéL
ULA";:INPUT A$
1650 IF A$="" THEN RETURN
1660 C1=ASC(A$)-64:IF C1<1 OR C
1>25 THEN 1640
1670 C2=VAL(MID$(A$,2)):IF C2<1
 OR C2>30 THEN 1640
1680 LOCATE 0,21:PRINTSPC(35):L
OCATE 0,21:PRINT"COPIANDO... AG
UARDE"
1690 IF TC=1 THEN 1970
1700 IF DC=1 THEN 1840
1710 IF C2<>S2 THEN 2090
1720 AW=ASC(MID$(D$(CC,CR),2))-
64:IF AW<>26 AND AW+C1-S1>25 TH
EN C1=25+S1-AW
1730 AW=ASC(MID$(D$(CC,CR),5))-
64:IF AW<>26 AND AW+C1-S1>25 TH
EN C1=25+S1-AW
1740 D$(S1,S2)=D$(CC,CR)
1750 FOR I=S1+1 TO C1
1760 A$=D$(I-1,S2)
1770 IF MID$(A$,2,1)="Z" THEN 1
790
1780 MID$(A$,2,1)=CHR$(ASC(MID$
(A$,2,1))+1)
1790 IF MID$(A$,5,1)="Z" THEN 1
810
1800 MID$(A$,5,1)=CHR$(ASC(MID$
(A$,5,1))+1)
1810 D$(I,S2)=A$:NEXT
1820 IF C1>CX THEN CX=C1
1830 GOTO 2080
1840 IF C1<>S1 THEN 2090
1850 AW=VAL(MID$(D$(CC,CR),3,2)
):IF AW+C2-S2>30 THEN C2=30+S2-
AW
1860 AW=VAL(MID$(D$(CC,CR),6)):
IF AW+C2-S2>30 THEN C2=30+S2-AW
1870 D$(S1,S2)=D$(CC,CR)
```

Tela de equações (à esquerda) e entrada de valores e rótulos no Acorn, micro sem equivalente no Brasil.

```
1880 FOR I=S2+1 TO C2
1890 A$=D$(S1,I-1)
1900 IF MID$(A$,2,1)="Z" THEN 1
920
1910 V=VAL(MID$(A$,3,2))+1:IF V
<10 THEN MID$(A$,3,2)=STR$(V) E
LSE MID$(A$,3,2)=MID$(STR$(V),2
,2)
1920 IF MID$(A$,5,1)="Z" THEN 1
940
1930 V=VAL(MID$(A$,6,2))+1:IF V
<10 THEN MID$(A$,6,2)=STR$(V) E
LSE MID$(A$,6,2)=MID$(STR$(V),2
,2)
1940 D$(S1,I)=A$:NEXT
1950 IF C2>RX THEN RX=C2
1960 GOTO 2080
1970 IF DC=1 THEN 2030
1980 IF C2<>CR THEN 2090
1990 FOR I=CC+1 TO C1
2000 D$(I,CR)=D$(I-1,CR):NEXT
2010 IF C1>CX THEN CX=C1
2020 GOTO 2080
2030 IF C1<>CC THEN 2090
2040 FOR I=CR+1 TO C2
2050 D$(CC,I)=D$(CC,I-1):NEXT
2060 IF C2>RX THEN RX=C2
2070 GOTO 2080
2080 MO=1:GOSUB 70:RETURN
2090 LOCATE 0,21:PRINTSPC(35):L
OCATE 0,21:PRINT"ERRO NA DESTIN
AÇAO":FOR D=1 TO 500:NEXTD
2100 GOTO 2080
```

```
1490  VTAB 23: HTAB 1: CALL  -
958: IF  LEN (D$(CC,CR)) = 1 TH
EN  PRINT "NADA PARA COPIAR"; C
HR$ (7);: FOR K = 1 TO 1000: NE
XT : RETURN
1495  PRINT "COPIA [A]BSOLUTA O
U [R]ELATIVA ";
1500  GET A$
1510  IF A$ =  CHR$ (13) THEN 2
085
1520  IF A$ = "A" THEN TC = 1:
GOTO 1540
1530  IF A$ <  > "R" THEN 1490
1535 TC = 0
1540  VTAB 23: HTAB 1: CALL  -
958
1550  IF TC = 0 AND  ASC (D$(CC
,CR)) <  > 131 THEN  PRINT "MOD
O DE COPIA INCORRETO - SO EQUAC
OES"; CHR$ (7);: FOR D = 1 TO 1
000: NEXT : GOTO 490
1560  VTAB 23: HTAB 1: CALL  -
958: PRINT "COPIA [C]OLUNA OU [
L]INHA ";
1570  GET A$
1580  IF A$ =  CHR$ (13) THEN 2
085
1590  IF A$ = "C" THEN DC = 1:
GOTO 1610
1600  IF A$ <  > "L" THEN 1560
1605 DC = 0
1610  VTAB 23: HTAB 1: CALL  -
958: PRINT "COMECA NA CEL.: ";:
 INPUT A$: IF A$ = "" THEN 2085
1620 S1 =  ASC (A$) - 64: IF S1
 < 1 OR S1 > 25 THEN 1610
1630 S2 =  VAL ( MID$ (A$,2)):
IF S2 < 1 OR S2 > 30 THEN 1610
1640  VTAB 23: HTAB 1: CALL  -
958: PRINT "TERMINA NA CEL.: ";
: INPUT A$
1650  IF A$ = "" THEN 2085
1660 C1 =  ASC (A$) - 64: IF C1
 < 1 OR C1 > 25 THEN 1640
1670 C2 =  VAL ( MID$ (A$,2)):
IF C2 < 1 OR C2 > 30 THEN 1640
1680  VTAB 23: HTAB 1: CALL  -
958: PRINT "COPIANDO - AGUARDE.
..";
1690  IF TC = 1 THEN 1970
1700  IF DC = 1 THEN 1840
1710  IF C2 <  > S2 THEN 2090
1720 A$ =  ASC ( MID$ (D$(CC,CR
),2)) - 64: IF A$ <  > 26 AND A
S + C1 - S1 > 25 THEN C1 = 25 +
 S1 - A$
1730 A$ =  ASC ( MID$ (D$(CC,CR
),5)) - 64: IF A$ <  > 26 AND A
S + C1 - S1 > 25 THEN C1 = 25 +
 S1 - A$
1740 D$(S1,S2) = D$(CC,CR)
1750  FOR I = S1 + 1 TO C1
1760 A$ = D$(I - 1,S2)
1770  IF  MID$ (A$,2,1) = "Z" T
HEN 1790
1780 A$ =  LEFT$ (A$,1) +  CHR$
 ( ASC ( MID$ (A$,2,1)) + 1) +
 MID$ (A$,3)
1790  IF  MID$ (A$,5,1) = "Z" T
HEN 1810
1800 A$ =  LEFT$ (A$,4) +  CHR$
 ( ASC ( MID$ (A$,5,1)) + 1) +
 MID$ (A$,6)
1810 D$(I,S2) = A$: NEXT
1820  IF C1 > CX THEN CX = C1
1830  GOTO 2080
1840  IF C1 <  > S1 THEN 2090
1850 A$ =  VAL ( MID$ (D$(CC,CR
),3,2)): IF A$ + C2 - S2 > 30 T
HEN C2 = 30 + S2 - A$
1860 A$ =  VAL ( MID$ (D$(CC,CR
),6)): IF A$ + C2 - S2 > 30 THE
N C2 = 30 + S2 - A$
1870 D$(S1,S2) = D$(CC,CR)
1880  FOR I = S2 + 1 TO C2
1890 A$ = D$(S1,I - 1)
1900  IF  MID$ (A$,2,1) = "Z" T
HEN 1920
1910 V =  VAL ( MID$ (A$,3,2))
+ 1: IF V < 10 THEN A$ =  LEFT$
 (A$,2) + " " +  STR$ (V) +  MI
D$ (A$,5): GOTO 1920
1915 A$ =  LEFT$ (A$,2) +  STR$
 (V) +  MID$ (A$,5)
1920  IF  MID$ (A$,5,1) = "Z" T
HEN 1940
1930 V =  VAL ( MID$ (A$,6,2))
+ 1: IF V < 10 THEN A$ =  LEFT$
 (A$,5) + " " +  STR$ (V) +  MI
D$ (A$,8): GOTO 1920
1935 A$ =  LEFT$ (A$,5) +  STR$
 (V) +  MID$ (A$,8)
1940 D$(S1,I) = A$: NEXT
1950  IF C2 > RX THEN RX = C2.
1960  GOTO 2080
```

O que é uma planilha integrada?

Os usuários de microcomputadores com maior disponibilidade de memória RAM — entre eles os da linha PC e alguns modelos da linha Apple — podem ter o privilégio de utilizar os programas integrados de planilha eletrônica. Eles recebem esta denominação por incluírem, além do programa de planilha de cálculo propriamente dito, outros programas, como os de elaboração de gráficos (a partir dos dados da planilha), montagem e gestão de bancos de dados relacionais, e, em alguns tipos, processamento de texto.

A grande vantagem de sistemas desse tipo é a perfeita integração de dados em cada uma das aplicações. Além disso, eles tornam dispensável trocar programas, quando se deseja utilizar um outro aplicativo.

Para os micros da linha Apple que têm memória RAM de 128 Kbytes ou mais, já existe um pacote integrado: o *Totalworks*. Mas os pacotes mais difundidos são os disponíveis para os micros da linha PC (de dezesseis bits), tais como o *Lotus 1-2-3*, o *Framework* e o *Symphony*.

```
1970  IF DC = 1 THEN 2030
1980  IF C2 <  > CR THEN 2090
1990  FOR I = CC + 1 TO C1
2000 D$(I,CR) = D$(I - 1,CR): N
EXT
2010  IF C1 > CX THEN CX = C1
2020  GOTO 2080
2030  IF C1 <  > CC THEN 2090
2040  FOR I = CR + 1 TO C2
2050 D$(CC,I) = D$(CC,I - 1): N
EXT
2060  IF C2 > RX THEN RX = C2
2070  GOTO 2080
2080 MO = 1: GOSUB 70
2085  VTAB 23: HTAB 1: CALL  -
958: RETURN
2090  VTAB 23: HTAB 1: CALL  -
958: PRINT "ERRO NA DESTINACAO"
: FOR D = 1 TO 1000: NEXT
2100  GOTO 2080
2500  IF  PEEK (222) = 5 THEN
GOTO 1470
2510  END
3000  HOME : PRINT "TERMINO O P
ROGRAMA? (S/N) ";: GET A$
3010  IF A$ <  > "S" THEN  GOSU
B 70: RETURN
3020  END
```

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

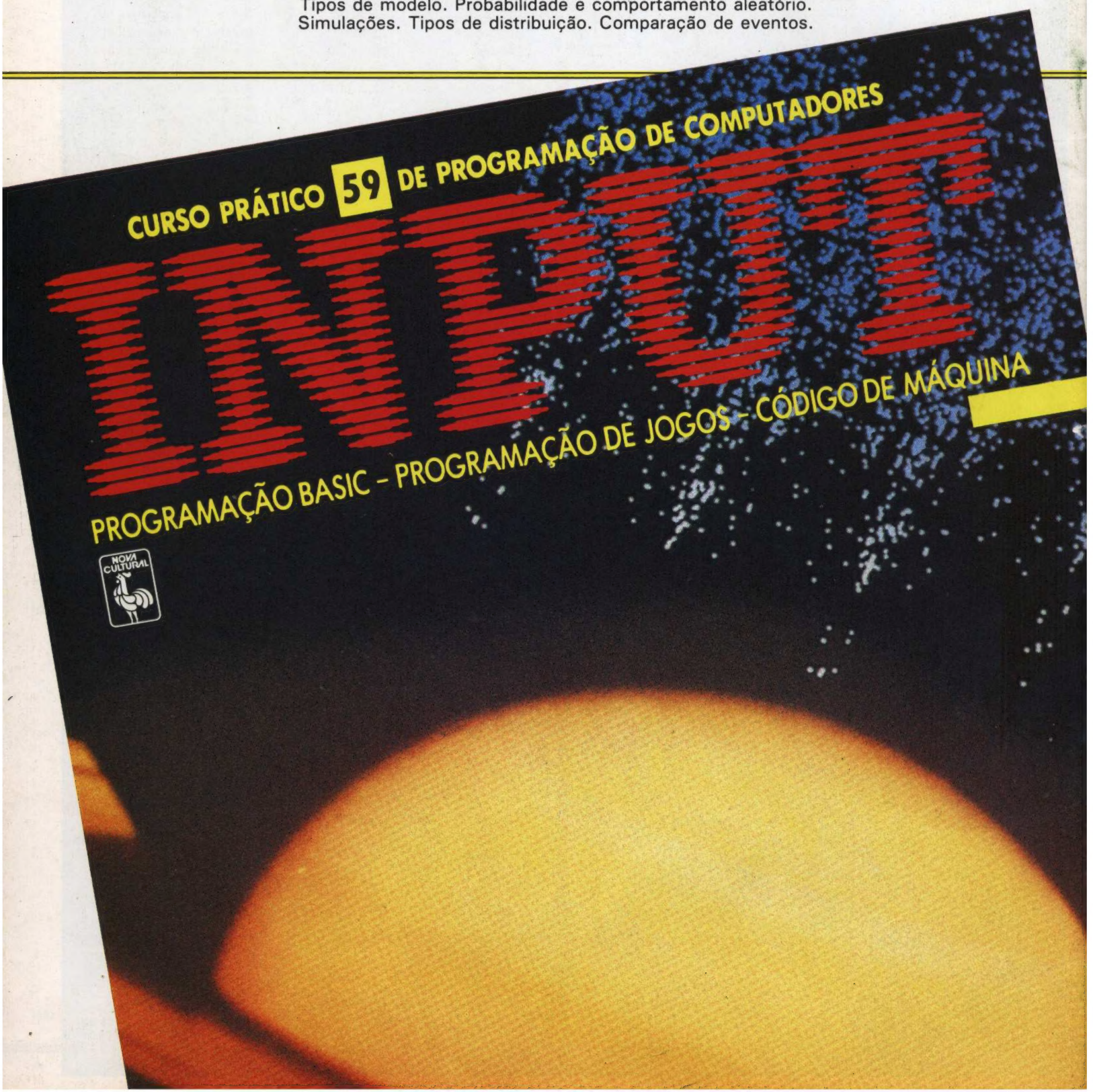
TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Famílias de curvas. Envoltórias. Focos e cúspides. Padrões de pontos. Ressonância. Caos. Dinâmica de populações.

CÓDIGO DE MÁQUINA
Avalanche: as quatro partes da rotina de salto. Efeitos sonoros. Ajustamento dos apontadores de tela.

PROGRAMAÇÃO BASIC
Tipos de modelo. Probabilidade e comportamento aleatório. Simulações. Tipos de distribuição. Comparação de eventos.